DESIGN AND EVALUATION OF A MULTIMODAL ASSISTIVE TECHNOLOGY USING TONGUE COMMANDS, HEAD MOVEMENTS, AND SPEECH RECOGNITION FOR PEOPLE WITH TETRAPLEGIA

A Thesis
Presented to
The Academic Faculty

by

Md Nazmus Sahadat

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2019

DESIGN AND EVALUATION OF A MULTIMODAL ASSISTIVE TECHNOLOGY USING TONGUE COMMANDS, HEAD MOVEMENTS, AND SPEECH RECOGNITION FOR PEOPLE WITH TETRAPLEGIA

Approved by:

Dr. Maysam Ghovanloo, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. David Anderson, Co-Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Omer Inan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Benjamin Klein
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Gregory Durgin
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Melody Jackson
School of Computational Science and Engineering
Georgia Institute of Technology

Date Approved: Mar 28, 2019

ACKNOWLEDGEMENTS

I would like to thank my family members, friends, and Lab colleagues for the continuous effortless support throughout my entire PhD life. Some name I would like to mention Nordine Sebkhi, Fanpeng Kong for effective and fruitful collaboration to contribute to scientific publications. I would also like to thank Sreyas Dighe, Nima Mikail, Pooja Srikhishnan and Arish Alreja for helping me design, test some part of this work. I would also like to thank my Advisor Dr. Maysaam Ghovanloo and Co-advisor Dr. David Anderson for constant support guidance to finish my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

People with high level (C1-C4) spinal cord injury (SCI) cannot use their limbs to do the daily life activities by themselves without assistance. Current assistive technologies (ATs) use remaining capabilities (tongue, muscle, brain, speech, sniffing) as an input method to help them control devices (computer, smartphone). However, these ATs are not very efficient as compared to the gold standards (mouse and keyboards, touch interfaces, joysticks, and so forth) which are being used in everyday life. Therefore, in this work, a novel multimodal assistive system is designed to provide better accessibility more intuitively. The multimodal Tongue Drive System (mTDS) utilizes three key remaining abilities (speech, tongue and head movements) to help people with tetraplegia control the environments such as accessing computers, smartphones or driving wheelchairs. Tongue commands are used as discrete/switch like inputs and head movements as proportional/continuous type inputs, and speech recognition to type texts faster compared to any keyboards to emulate a mouse-keyboard combined system to access computers/ smartphones. Novel signal processing algorithms are developed and implemented in the wearable unit to provide universal access to multiple devices from the wireless mTDS. Non-disabled subjects participated in multiple studies to find the efficacy of mTDS in comparison to gold standards, and people with tetraplegia to evaluate technology learning abilities. Significant improvements are observed in terms of increasing accuracy and speed while doing different computer access and wheelchair mobility tasks. Thus, with sufficient learning of mTDS, it is feasible to reduce the performance gap between a non-disabled and a person with tetraplegia compared to the existing ATs.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Every year ∼12,500 new cases of spinal cord injuries (SCI) are added to a population of 276,000 individuals in the United States alone [1]. 47% of this population have tetraplegia can neither use their legs nor their arms [2]. 92% of those with SCI live in their private residences after recovery, relying on family members or professional caregivers to do daily life tasks [2]. The estimated medical and caring cost of tetraplegia as a result of SCI at level (C1-C4) for the first year is on average $1,064,716 and $184,891 in subsequent years, which is a substantial burden on the individual as well as national budgets [3].

Assistive Technologies (ATs) help people with physical disabilities using their remaining abilities to obtain autonomy in everyday tasks, such as accessing a computer, driving a wheelchair, mitigate healthcare costs and most importantly, improving the quality of life. Different efforts have been underway to help people interact with computer using ATs, such as non-invasive BCIs [4], Invasive BCIs [5], EMG [6], eye tracker [7], speech control [8], head tracking [9], sniffing [10], and head control switches [11]. Apart from usability problems in the form of the "Midas touch," which is referred to issuing commands that are not intended by the user [12], cosmetic appearances, slow response times due to limited bandwidth, low reliability as a result of susceptibility to noise [13], muscle fatigue from chronic use of EMG based systems [14], none of the devices in these AT categories has been able provide anything close to the speed and ease of access that keyboard-mouse combination (KnM) can offer to able-bodied individuals.

Tongue-controlled devices seem to perform better in terms of susceptibility to noise, muscle fatigue, and issuance of false positive commands because of some of the inherent abilities of the human tongue [15, 16]. However, in comparison to KnM, they are slow and limited. Tongue-Touch Keypad [17], Inductive Tongue Control System [18], Tongue Mouse

[19], Tongue Point [20], Jouse3 [21], optical tongue gesture sensing system [22], and Tongue Drive System (TDS) [23], are a few examples of tongue-based ATs. Dual-mode Tongue Drive System (dTDS) [24], which combined speech recognition with tongue commands took advantage of the versatile speech recognition technology to provide considerably faster typing capability, but it did not offer any proportional control capability for cursor navigation. Lack of simultaneous multimodal accessibility fails to provide features like drag and drop, anchoring, multiple object selection and many other features that need multiple inputs from the end user at the same time.

A multimodal assistive technology in the form of a wearable wireless system with various sensors, capable of capturing several gestures to provide the user with multiple mutually-exclusive inputs, simultaneously could be a potential solution to this problem.

1.2 Related Work

Tetraplegia is the result of damage to the spinal cord at the upper cervical level (C1-C5) and typically results in varying degrees of paralysis in the upper limbs [25]. However, movement of the muscles at the neck level and above are often preserved and utilized as remaining abilities for navigating wheelchairs, accessing computers (or smartphones), and controlling the user environment. Assistive Technologies (ATs) facilitate the use of these remaining abilities, such as voluntary muscle movements [26], eye tracking [7], [27], [28], tongue motion [17], [20–22], [24], [18, 29–31], head tracking [9], [32–34], as well as speech [35, 36], sniffing [10], and brain signals [4, 5, 25] for daily activities.

Some remaining abilities, such as brain signals and speech, are only suitable as a control mechanism in specific conditions when the environment is less noisy [16]. Head movements, on the other hand, works well as a proportional control input in most conditions, but it builds up neck muscle fatigue and may be affected by inertial forces and artifacts in a moving vehicle or wheelchair navigation in rough terrain. Tongue-operated ATs have shown superior performance in terms of robustness against noise and muscle fatigue in long-term usage thanks to inherent abilities of the human tongue [16], [17], [20–24], [18, 29–31]. They are most suitable for discrete switch type control, as demonstrated in several studies [23], [37].

To operate a complex device like a computer efficiently in any modern operating system, able-bodied users need both discrete (keyboard and mouse click) and continuous-proportional (cursor) inputs at their disposal. These human input mechanisms are usually used either sequentially (e.g., moving the cursor to an icon and click on it) or in a parallel manner (e.g., hold the mouse button down to drag and drop an icon) [38]. Almost every consumer level human-computer interaction (HCI) system, from desktops to smartphones, is equipped with human interface devices (HID) that provides both discrete and proportional control inputs, such as keyboard, mouse, touchscreen, and touchpad. However, most ATs today are incapable of offering both continuous and discrete inputs, and even less so simultaneously. As a result, end users with physical disabilities, utilizing these ATs, are either too slow or completely incapable of accessing computers efficiently, particularly in doing more complex tasks.

Therefore, to enable individuals with paralysis, particularly those who are unable to use their upper limbs, e.g., tetraplegia, to access computers with a flexible set of modalities, which can utilize their remaining abilities by engaging different organs, to which they maintain voluntary control, is imperative. The nature of this interaction should add both discrete-switch based and continuous-proportional inputs, to leverage and build upon the existing mechanisms that are readily built in every modern operating system for able-bodied users. As an added benefit, it would be strongly preferred if these modalities are available simultaneously to be used either sequentially or in a parallel manner based upon user's choice or need.

In the following sections of this chapter, some of the existing tongue controlled ATs with different design principles will be discussed.

1.2.1 Tongue Touch Keypad (TTK)

Tongue Touch Keypad (TTK) uses the pressure of the tongue tip to activate one of the nine pressure sensitive switches placed on the palate of the oral cavity [17]. The electronics is molded with custom acrylic to avoid saliva leaking into the device. Activated switch command is transmitted wirelessly to the receiver. The receiver translates the command to

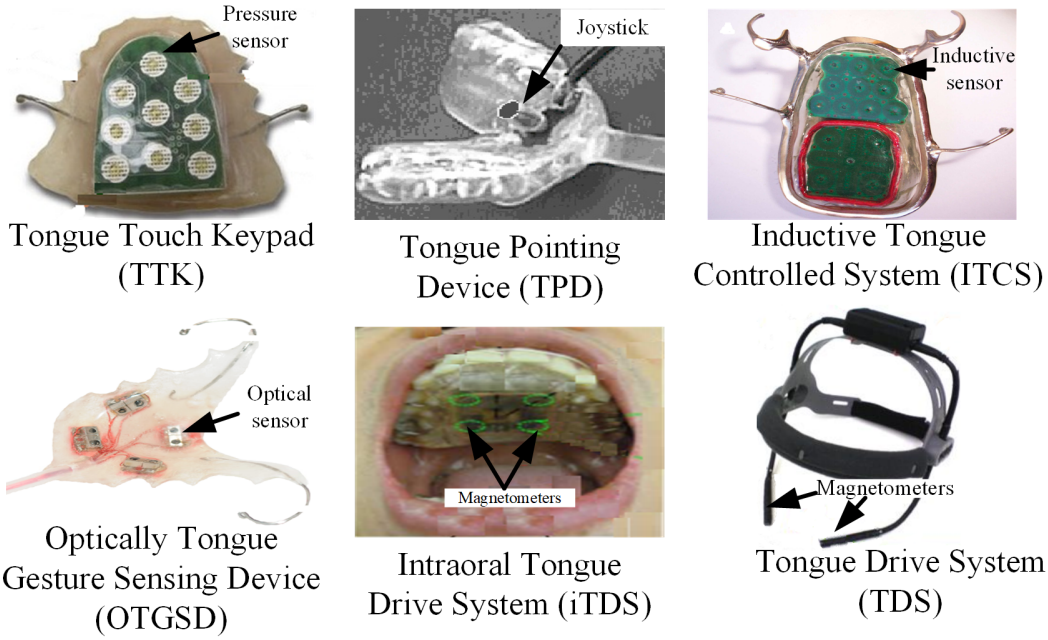

Figure 1: (a) Tongue Touch Keypad (TTK) (b) Tongue Pointing Device (c) Inductive Tongue Control System (d) Optically Tongue Gesture Sensing Device (e) intraoral Tongue Drive System (f) Tongue Drive System.

control signal to drive a wheelchair or to control a computer [30]. Figure 1 shows the TTK with custom acrylic molded pressure sensitive keypad.

1.2.2 Tongue Pointing Device (TPD)

Tongue pointing device [20] uses IBM TrackPoint technology to generate commands. This device is used to move the cursor proportionally based upon the displacement of the joystick created by the tongue. The 1 cm long TrackPoint shown in Figure 1 might be uncomfortable for the users while talking or eating.

1.2.3 Inductive tongue controlled system (ITCS)

Inductive tongue computer interface (ITCI) is developed by the researchers at Alborg University in Denmark [18], can detect the change of inductance by placing a ferromagnetic material attached to the tongue on an array of inductive switches [39]. The planar inductors are implemented on a PCB. As a result, an AC voltage change is detected from the change of inductance when the tongue tip is placed in it. The system uses 18 inductor switches [40] and can be used to drive a wheelchair [41], control a computer [42]. Figure 1 shows the

device with custom-molded clasps to secure it on the teeth.

1.2.4 Optically Tongue Gesture Sensing Device (OTGSD)

The OTGSD uses optical sensors to detect the tongue gestures. This system is designed by the researchers at University of Washington [22]. It uses four infrared light emitting diodes (LEDs) and photodiodes in the mouth to find the tongue position when approaches to the sensors by processing the amount of reflection received by the photodiode shown in Figure 1. The device delivers data using wired communication to process commands in the PC. This technology is prone to issuing many unwanted commands when the tongue moves during speaking or any other natural motions such as coughing, sneezing, swallowing, and chewing.

1.2.5 Tongue Drive System (TDS)

Tongue Drive System (TDS) uses magnetic field generated by a magnet tracer attached to the user's tongue to detect commands issued by a set of pre-defined voluntary tongue gestures [29]. Four tri-axial magnetic sensors, mounted on a pair of lateral poles of a headset, are held near the cheeks to capture the magnetic field variations resulted from the movements of a tracer attached on the tongue [29]. Control unit packetizes and transmits the sensor data wirelessly at 2.4 GHz to a receiver [43], [44]. The receiver unit receives the magnetic data packets and sends to a PC or smartphone [44] to process and find the assigned commands [45]. Figure 1 shows a complete system, which can drive a wheelchair, control smartphone, and PC [44]. Figure 1 shows the intraoral version of the Tongue Drive System (iTDS) by placing the TDS electronics in the mouth. iTDS has a better signal to noise ratio because the sensors are placed closer to the magnetic tracer inside of the oral cavity. The system has better aesthetic appearance since it is hidden in the mouth [46]. The sensor data is transmitted wirelessly by a dual-band, 27 MHz and 433 MHz, radio frequency (RF) transmitter to a PC or smartphone via receiver similar to the eTDS. The main drawback of iTDS is occupying a small portion of the intraoral space and somewhat limiting the tongue movement in the mouth.

1.2.6 Others

Jouse3 [21] is a joystick-based AT, controlled by the tongue. It is usually mounted on any desk, wheelchair, or bedframe, and used to control computers and other mobile devices. This device can only be operated when the user is not in motion.

A piezoelectric sensor based tongue control device is also developed to control PC [19]. The user bites on the device, and move the tongue on the sensor plate to navigate the cursor like a touchpad. The piezoelectric sensor matrix measures the pressure created by the tongue to find the tip position. The large form factor is a major drawback of this system. This device might not be feasible to use while talking or eating.

1.3 Research Objectives

The objective of this research is to design a wireless and wearable multimodal assistive technology that will utilize multiple remaining abilities of the people with physical disabilities, such as tetraplegia, to do their daily life activities more efficiently and independently, thus improving their quality of life. The focus of this work has been designing a system, called multimodal Tongue Drive System (mTDS), that can capture speech, tongue commands, and head movements and convert them to text and commands to interact with devices such as computers, smartphones, and wheelchairs. High accuracy signal processing algorithms will be developed to process user inputs which are efficient to compute in a limited resource embedded hardware. The algorithms will also be optimized to reduce power consumption and increase battery lifetime. This work will also focus on designing experiments to evaluate the pros and cons of the multimodal interactions of the proposed assistive technology (AT) in effectively utilizing the remaining abilities of the user, and apply them to accessing computers, using smartphones, and driving wheelchairs intuitively and efficiently. The evaluation study will include both the healthy able-bodied and tetraplegic individuals. Able-bodied individuals will allow us to compare the mTDS performance against default and gold standard interfaces, i.e. keyboard-mouse combination, touchscreen, and joystick, for accessing computers, smartphones, and wheelchairs, respectively. Tetraplegic individuals, on the other

hand, will allow us to evaluate the mTDS performance by the intended population and collect valuable feedback from the end users, caregivers, clinicians, and care providers. As the ultimate outcome of the research we intend to test the hypothesis that by providing multiple simultaneous control abilities which are still accessible to the end users, one can significantly reduce the performance gap between healthy and tetraplegic individuals in controlling and accessing three key target devices that are necessary in today's daily living tasks: computers (for work, education, entertainment, etc.), smartphones (for communication, environment control, seeking assistnace, etc.), and wheelchairs (for mobility). The following tasks defines the scope of this research:

Task 1: Designing a wireless and wearable system (mTDS) that can simultaneously capture speech, tongue motion, and head movements as inputs to control the target devices.

Task 2: Design, implementation, and testing of new algorithms and firmware for a standalone version of mTDS that can process both tongue and head movements real time in the wearable headset.

Task 3: Evaluate the mTDS by the healthy able-bodied individuals in conducting computer tasks and compare with gold standard, keyboard-mouse combination and previously developed system, TDS.

Task 4: Evaluation of the mTDS in doing computer access, smartphone access, and wheelchair navigation tasks by the people with tetraplegia.

CHAPTER 2

DEVELOPMENT OF MULTIMODAL TONGUE DRIVE SYSTEM (MTDS)

The complete development process is presented in three phases. In phase I, all the sensor signals are captured by a wireless headset and sent to the computer to process assigned commands to control it. In phase II, the signal processing is done by an embedded system to process commands and control a wheelchair or PC. In phase III, all the signal processing will be done in a wireless headset which can generate HID commands from the assigned gestures to control a PC, smartphone or a wheelchair seamlessly.

2.1 System Architecture: Phase I

The mTDS hardware is composed of two parts, Part-A: a wearable headset including magnetic sensors to capture tongue gestures, inertial sensor to capture head orientation, and a microphone to capture voice, Part-B: a USB receiver dongle to receive magnetic and inertial sensor data and a Bluetooth receiver to receive audio signal. Figure 2 shows a simplified block diagram of the system to help explain how multiple sensor data is converted to discrete and continuous output to provide simultaneous multimodal computer access. Figure 3 provides a simplified schematic of both Part-A and Part-B. Part-A is divided to headset assembly, sensor array, and control unit, while Part-B is described as the audio subsystem and USB receiver dongle.

2.1.1 Headset Assembly

The wearable headset is constructed on a headgear with additional 3D-printed assemblies, housing a magnetic sensor array, a control unit and a commercial Bluetooth microphone, as shown in Figure 4. The 3D printed box on top of the headset has two chambers, a smaller one on top houses the electronics, while a bigger one below it contains the batteries. Two 3D printed arms are also mounted along the sides of the headgear using plastic screws. A

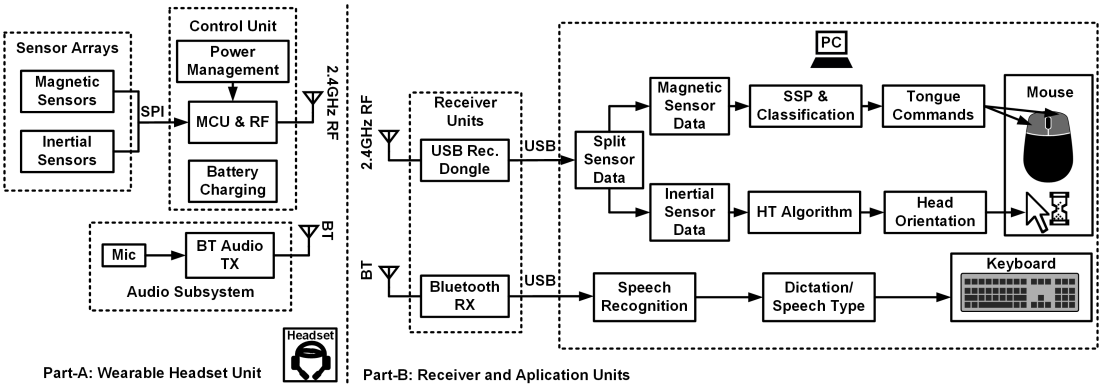

Figure 2: A wearable headset Unit (Part- A) captures the tongue tracer movement using magnetic sensor, head tracking using inertial sensor and speech using microphone, sends the data to RF and Bluetooth receiver (Part-B). Received magnetic sensor data translated to tongue commands to emulate mouse clicks after sensor signal processing and classification, inertial sensor data translated to capture head orientation to do proportional mouse movement and audio signal to speech type/ dictation as a keyboard.

housing for the magnetic sensor array slides in each arm, allowing it to be adjusted in 6 positions to place the sensor array near the user's cheeks. The housing also has guides for flat cables connecting the magnetic sensors to the control unit. The Bluetooth microphone is placed at the tip of the left magnetic sensor array.

2.1.2 Sensor Array

Two 4-layer $38.4 \times 7.1mm^2$ sensor printed circuit boards (PCB), each containing two 3-axis accelerometers and 3-axis magnetometers (LSM303D, STMicroelectronics, Switzerland), are mounted in each sensor arm (Figure 4). Each sensor arm is mounted on one side of the headset, and the sensor boards are connected to the control unit via a 10 pin, 1.27 mm pitch, flat cable (Samtec Inc., New Albany, IN).

2.1.3 Control Unit

The control unit, implemented on a 4-layer PCB ($25.4 \times 12.7mm2$), contains power management block, battery charging circuit, microcontroller unit (MCU), a 9D inertial measurement unit (IMU), programming/debugging headers, and sensor headers. Signal traces are shielded to reduce interference. The control unit uses an inverted-F wiggle PCB antenna to transfer data. To optimize the antenna performance, metal underneath the antenna block is removed

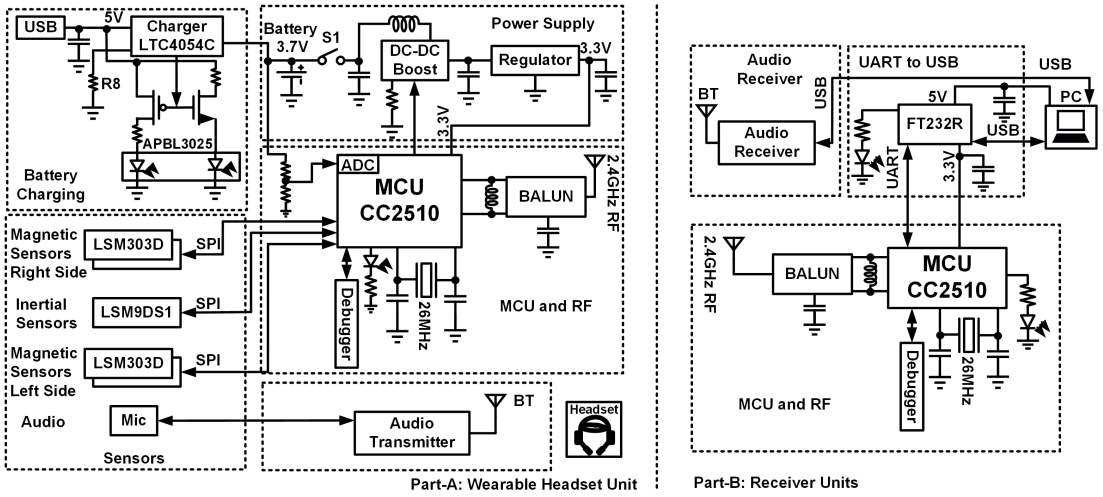

Figure 3: Simplified schematic of the Wearable Headset Unit (Part-A) and receiver units (Part-B). RF 2.4GHz transceiver transmits the magnetic and inertial data from the headset control unit, receives by USB receiver dongle. A Bluetooth link is used to transmit audio signal.

from Layers 2, 3 and 4. An electroless nickel immersion gold (ENIG) finish is applied to reduce the effects of oxidation.

The power management block in Figure 3 includes an on/off switch (S1), DC-DC boost converter (TPS61220DCKR), and LDO (TPS71733) to extend battery life. When the battery is full, LDO powers the control unit and sensors, and the boost converter is kept off to increase power conversion efficiency. When battery voltage drops below a predefined threshold, 2.5V here, the boost converter is enabled by the MCU to allow the control unit continue its operation till the battery voltage falls below 0.7 V. The charging circuitry includes a linear charging chip (LTC4054), which charging current is controlled by a resistor, R8. This unit receives power from a 5 V micro-USB port, and is capable of charging a variety of Li-ion batteries. Red and Green LED indicators indicate the charging status.

The MCU (CC2510, Texas Instruments) is clocked at 26 MHz by an external crystal oscillator. Its built-in transceiver unit is configured to transmit/receive data at 2.45 GHz up to 500 Kbaud. Positive and negative Radio Frequency (RF) traces are differentially routed (8 Mils trace) to a balun (2450BM15B0003, Johanson technology, Camarillo, CA) for impedance matching followed by a 22 pF capacitor and 10 nH inductor, before feeding to the PCB antenna.

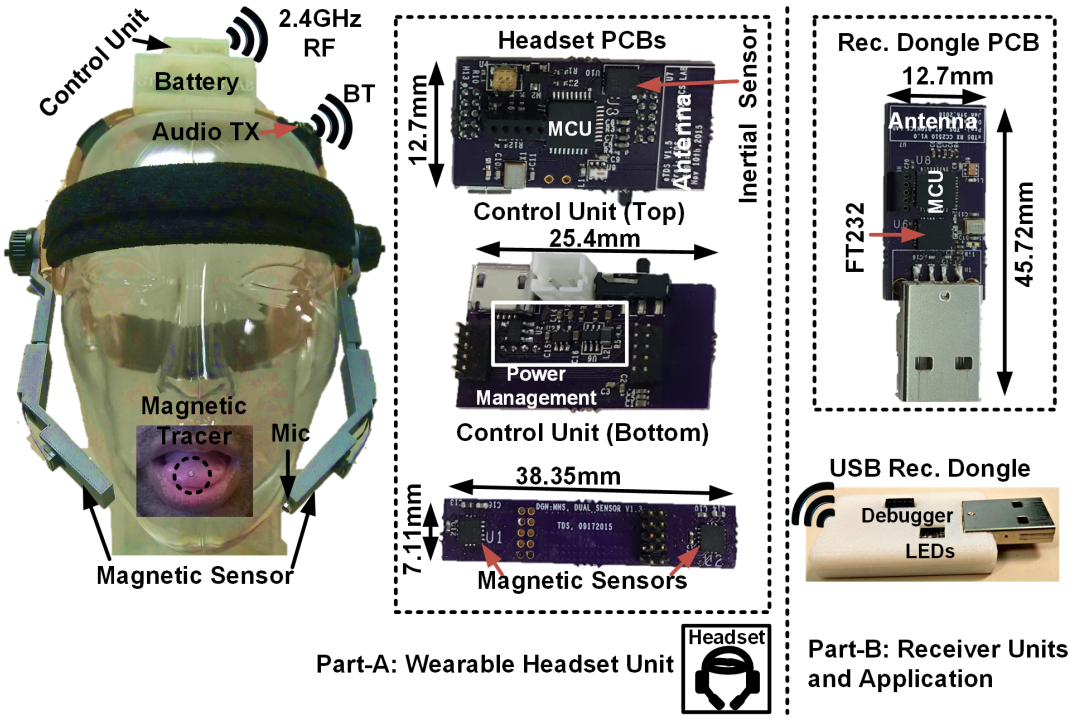

Figure 4: Simplified schematic of the Wearable Headset Unit (Part-A) and receiver units (Part-B). RF 2.4GHz transceiver transmits the magnetic and inertial data from the headset control unit, receives by USB receiver dongle. A Bluetooth link is used to transmit audio signal.

An additional 9D IMU (LSM9DS1, STMicroelectronics) is mounted on the control unit PCB, as far as possible from the magnetic tracer. All sensors are configured to operate at a sampling frequency of 100 Hz, representing data samples as 16 bit unsigned integers. They communicate with the MCU over a shared serial peripheral interface (SPI) bus at 115.2 Kbps, using different chip selects.

2.1.4 Audio Recording

A commercial Bluetooth audio transmitter (Jabra, Denmark) captures user voice input, while being powered from the same battery as the rest of the mTDS, but turned on/off by a separate switch. A small microphone (POM-3535L-2-R, PUI Audio, Inc.) is placed in the left sensor array assembly, as shown in Figure 4.

2.1.5 USB Receiver Dongle

A custom designed USB receiver, shown in Figure 4, housed in a 3D printed dongle facilitates communication between a PC and the control unit. The dongle includes a MCU (CC2510) clocked at 26 MHz, a USB controller (FT232RQ, FTDI) and a balun (2450BM15B0003, Johanson technology, Camarillo, CA) to enable the use of a monopole inverted F wiggle PCB antenna (Figure 3). Two LEDs show the data transfer status (Transmit/Receive) of the dongle. Communication with the PC is achieved via a virtual COM port, configured at a baud rate of 921.6 Kbps. A 5-pin header offers the ability to apply firmware updates.

The mTDS firmware is explained in two subsections, one for the control unit to transfer magnetic and inertial sensor data and one for USB receiver dongle to receive and deliver to a PC.

2.1.6 Control Unit Firmware

The control unit firmware follows a multi-stage initialization sequence which begins with bootstrapping critical system components included the clock oscillator, General-purpose input/output (GPIO), SPI communication, sleep timers and the Analog to Digital Converter (ADC). This is followed by sensor configuration and initialization, such as 100 Hz sampling rate and dynamic ranges for accelerometers ($\pm 2g$), magneto-meters ($\pm 4gauss$), and gyroscopes ($\pm 250°/s$). Finally, the RF transceiver is configured to initiate a handshake with the USB receiver dongle and establish connectivity. The handshake involves 5 repeated broadcasts of a pilot packet train, while awaiting receipt and validation of an acknowledgement packet. If a handshake is not completed after 60 broadcasts of the pilot packet train, the control unit enters a power saving mode.

When connectivity is successfully established, the control unit updates the standby threshold duration values based on information received from the USB dongle. In standby mode, the MCUs sleep timer frequency is reduced to execute the state machine at $1Hz$, and remain in a power saving mode at all other times. Outside of standby mode, the state machine executes at $100Hz$. While transmitting data, the control unit can be put into standby mode, based on instructions received from the USB receiver dongle. Once configuration is

completed, the control unit enters a data transmission state, and transmits battery voltage as well as data from the 4 magnetometers near the cheeks, the accelerometer and gyroscope in the control unit. These readings are placed in a single packet, which is also indexed with a packet counter to track missing packets. Interrupt based direct memory access (DMA) is used to ensure reliable data transfer, and move the RF data to a global packet buffer.

2.1.7 USB Receiver Dongle Firmware

The receiver dongle firmware executes an initialization process by configuring the clock oscillator, Universal Asynchronous Receiver/Transmitter (UART), GPIO, DMA, timer, and RF registers to match the transmitter's carrier frequency and baud rate. After initialization, the dongle awaits a "connection request" from the PC. Once a PC requests a connection with an available headset, the USB dongle starts listening for handshake packets broadcast by the mTDS control unit. It waits until it receives a handshake packet and responds to the mTDS headset with an acknowledgement to complete its handshake. It sends a similar acknowledgement packet to the PC to signal successful connectivity. This allows the PC to configure the dongle to start sending requests for data to the mTDS headset. Data received from the mTDS headset's control unit is examined for packet validity, and passed up to the application running on the PC. A watchdog timer enforces the timeout limits associated with the Transmit/Receive process, resetting it when necessary.

2.2 System Architecture: Phase II

The goal of designing this system is to process all the sensor signals in an embedded hardware to find user's gestures and remove the dependency on a PC or smartphone. Processed gestures are translated into commands to interact with both computer and power wheelchair. The architecture of the system is shown in Figure 5. The following sections will describe the different blocks of processing unit.

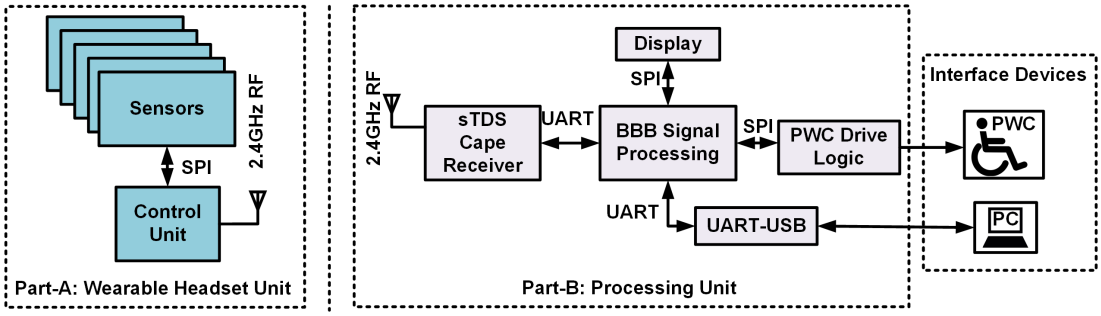

Figure 5: Simplified block diagram of the system which uses processing unit (part-B) to generate commands to navigate wheelchair and control PC. control unit, magnetic sensor and receiver PCBs are also shown in the figure.

2.2.1 BeagleBone Black

The BeagleBone Black (BBB), single board computer (SBC) is used because of the low power consumption, open source and adaptability with different wireless and display modules according to the need of the system. It also has enough computation capabilities to calibrate, train, and execute the proposed signal processing and machine learning algorithm. Figure 6 shows simplified schematic diagrams of parts B. The heart of part B, which is in charge of the embedded TDS processing, is a BBB, a cost-effective credit card sized SBC with a 1 GHz ARM Cortex A8 processor with open support community. It has 512 MB RAM, 4 GB onboard flash storage, a USB client for power and communication (SSH/SCP/USB), a USB host, and two 46-pin extension headers, which create a powerful substitute for the smartphone used for processing in the earlier version of the TDS [47]. An open-source Linux based operating system, Angstrom kernel v3.8 in this prototype, runs on the BBB. The extension headers, which contain 5 V and 3.3 V supply pins, general-purpose input/output (GPIO), universal asynchronous receiver/transmitter (UART), and SPI connectivity, are used to add commercial or custom electronics in the form of piggy-back boards, known as capes. Here, we have designed a custom cape for the embedded TDS to provide regulated power from the wheelchair battery to BBB, display, wireless connectivity with the TDS headset (part A), Wi-Fi and Bluetooth connectivity with other devices in the user environment, wheelchair control and drive circuitry, and PC interfacing logic.

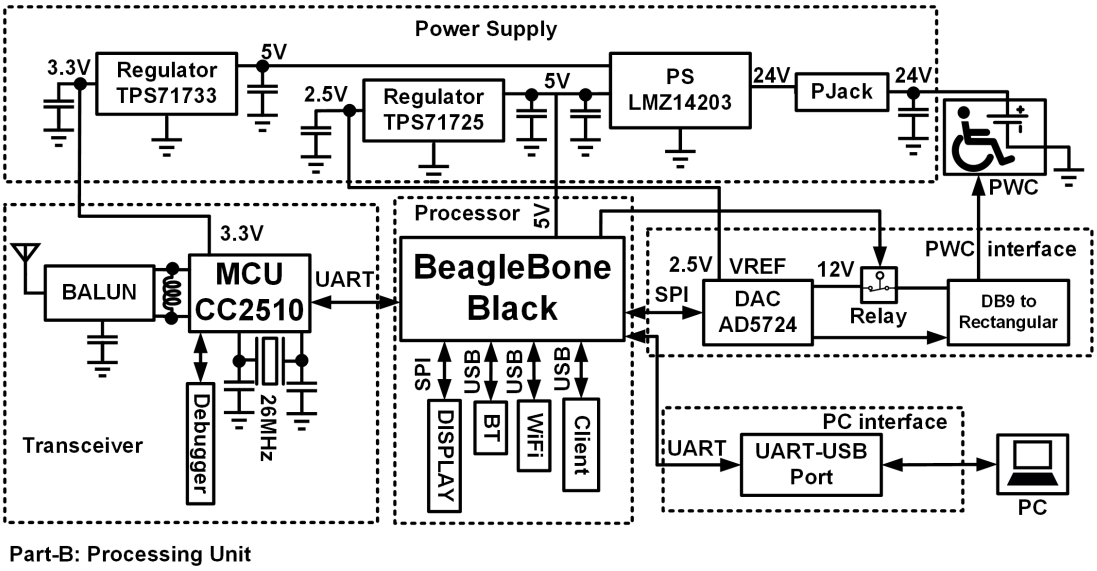

Figure 6: Processing Unit Simplified Schematics

2.2.2 Power Supply

We have utilized the powered wheelchair batteries to supply part B, because TDS end users spend most of their awake time on their wheelchairs. The powered wheelchair used to develop the current prototype was a Quantum 6000 (Q6000) wheelchair donated by Pride Mobility Inc. (Exeter, PA) [48]. This wheelchair is equipped with a pair of 12 V deep cycle absorbent glass mat (AGM) batteries that are connected in series to provide 24 V, which is accessible through its XLR 3 pin male type connector from the wheelchair charging port. Hence, 24 V is the cape PCB input via a barrel power connector. The power supply unit generates 5 V (3 A), 3.3 V, and 2.5 V supplies for various components of the embedded system from the 24 V supply, using LMZ14203, TPS71733, TPS72725 (TI, Dallas, TX), respectively, as shown in Figure 7.

2.2.3 Wireless Interfaces

The custom cape is equipped with a 2.4 GHz RF transceiver, built-in with the same MCU (CC2510) used in the TDS headset.These two chips use TI's proprietary RF protocol to deliver sensor data packets from part A to part B. This unit also includes a matching circuit and a BALUN to convert the CC2510 differential output to a single-ended 50Ω inverted-F

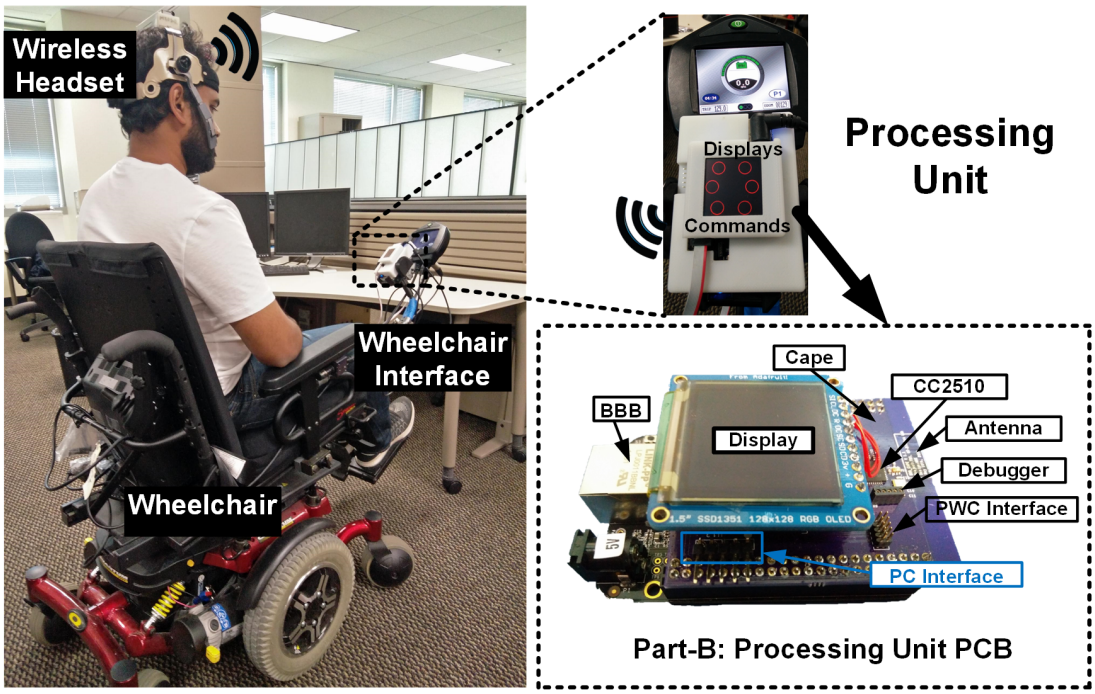

Figure 7: Processing Unit PCB with the system

antenna (IFA) [49], which is implemented on the cape PCB. As shown in Figure 7, the wirelessly-received packet is delivered from the MCU to BBB via a UART link.

A commercial 150 Mbps Wi-Fi dongle (EDIMAX, Taiwan) is plugged into the BBB USB host to add Wi-Fi connectivity to the mTDS.

2.2.4 Display

Part B is equipped with a $128\times128 pixel$ organic light-emitting diode (oLED, $26.8\times26.8 mm^2$, SSD1351) display (Solomon Systech Ltd., China), shown in Figure 7, to provide the end user with visual feedback on the current tongue commands being executed. The display control commands are sent from the BBB using the SPI interface. The command display frame update rate is configured at 1 Hz in order to not occupy too much of the BBB processor time.

2.2.5 Powered Wheelchair Control

A 9-pin D-sub (DB-9) connector is used to interface between the embedded TDS part-B and Q6000 wheelchair through its standard Q-logic interface for alternative control, which

is specifically designed for navigation controllers other than the default joystick [50]. This interface controls the wheelchair motion by adjusting two analog vectors that change from 2 V to 10 V to set the linear (forward-backward) and rotational (left-right) speed of the wheelchair. The wheelchair interface circuitry uses a 12-bit serial input quad output DAC (AD5724) to generate the analog vectors. Three out of four DAC outputs are used to generate two vectors (linear and rotation) and a neutral voltage (6 V). If both DAC channel voltages (linear/rotation) are set to 6 V, then the wheelchair will not move in any direction. A value of more or less than 6 V is set to change the linear, and rotational vectors to move the wheelchair forward or backward and left or right. This neutral voltage value can be adjusted by reprogramming the wheelchair. The interface between the BBB and DAC is via the same SPI port used for display, using a different chip select. A safety mechanism is added to the wheelchair interface using relays to disarm the DAC power supply, controlled by a BBB GPIO pin, as shown in Figure 7. This mechanism allows the user to disable the wheelchair from the user interface software, in addition to the hardware emergency stop, which is a mechanical push button that is used to entirely shut down the wheelchair via its built-in safety mechanism, if necessary.

2.2.6 PC Interface

We have used a UART to USB converter, in the form of a commercially-available USB to serial TTL cable, to interface between the PC and the embedded TDS part B, as shown in Figure 7. The baud rate of this communication is set to 921,600 bits/s, and a UART header is added to the custom-designed cape to facilitate this functionality.

2.2.7 Software Architecture

The online execution phase consists of two functions: one on computer access and the other on wheelchair navigation, which are arguably the most important daily tasks for most TDS users [23]. This real-time SSP algorithm, which includes SVM-based pattern recognition, is implemented in BBB using C++. Support vector machine with a linear kernel is used to classify seven tongue commands because of the high accuracy, low computation requirements, and low false positive rate compared to the KNN algorithm used in the previous system

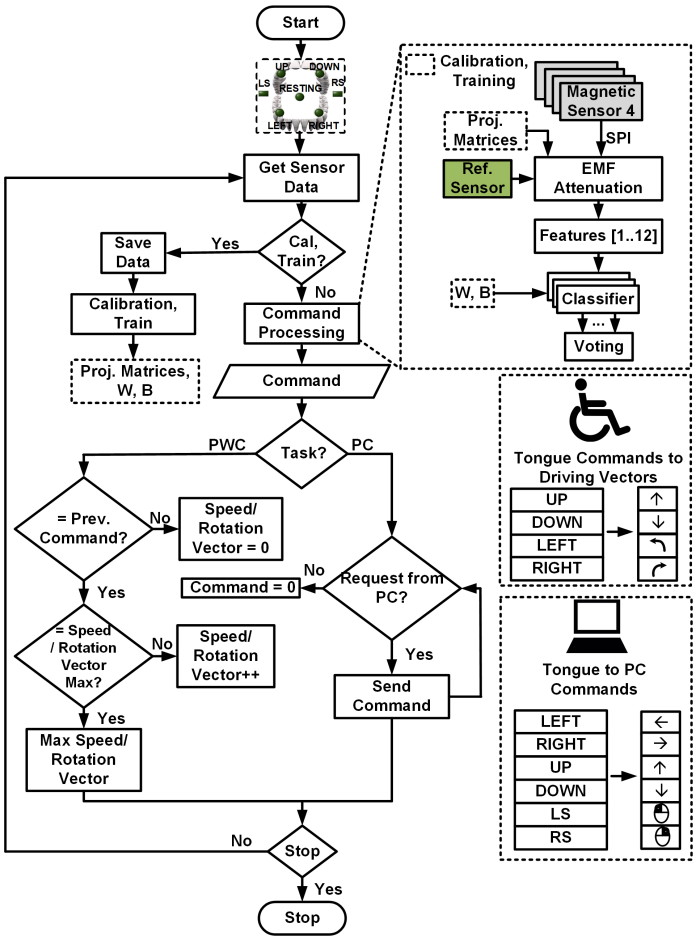

Figure 8: Simplified flowchart showing TDS communication with PC and wheelchair, as well as the vector-based linear and rotation speed control of the wheelchair. Real-time command processing step are shown in the right.

[51]. The system requires a 20 s calibration and 70 s training (10 s for each command) by placing magnetic tracer to the shown landmarks (different teeth and cheek) in Fig. 3 before execution. Data contains outputs from the five sensors (four of them around the cheek) including a reference magnetometer (control unit). A least square error (LSE) method is used to find four projection matrices from the calibration data to attenuate the earth magnetic field (EMF) interferences. 12 dimensional EMF denoised features are used as inputs to the one vs. one classification algorithm during the training and execution. 21 support vector machines (7 tongue commands) are trained to find the weights, W and bias, B matrices. During the execution, maximum vote wins technique is used to find the winning class among 7 commands as an output. All these computations were done in the BBB at

Table 1: Benchmarking the existing tongue-controlled assistive devices

Features	TTK [17]	OSTG [22]	ITCS [18]	TDS [23]	iTDS [31]	This work [52]
Processor	MCU	PC	CU	PC/ Phone	PC/ Phone	ARM A8
Standalone	yes	no	yes	no	no	yes
# of wireless link	1	1	1	1	1	3
Connectivity	RF	RF	RF	RF	RF	RF, BT, Wi-Fi
Internet	no	no	no	no	no	yes
Remote emergency navigation/ monitor	no	no	no	no	no	yes
Display / Feedback	no	no	no	no	no	yes
In system calibration/ training	no	no	no	no	no	yes
Obtrusive	no	no	no	yes	no	yes
# of commands	9	4	18	7	7	7
Avg. driving velocity (m/s)	-	-	0.18∼0.29	0.25*	-	0.22∼0.34
Avg. Driving errors	-	-	0∼3	2.1*	-	0∼7
Command activation	Pressure	Optical switch	Inductive switch	Virtual switch	Virtual Switch	Virtual switch

TTK: Tongue Touch Keypad, TDS: Tongue Drive System, iTDS: intraoral Tongue Drive System, OSTG: optically sensing tongue gestures, ITCS: inductive tongue control system, MCU: microcontroller unit, RF: radio frequency, BT: Bluetooth, CU: central unit. * mean value of all subjects

a sampling rate of 100 commands/s. A state machine shown in Figure 8, implemented to select the device under control (PC or PWC). If PC is selected, then tongue commands are translated to keyboard arrow keys to navigate the cursor to play the game. For PWC control, tongue commands are translated to analog driving vectors as DAC outputs. This process continues in an infinite loop until there is a stop request by the user through the UI.

The current system does not use a microphone to detect the speech of the user. However, the algorithm utilizes tongue movements to differentiate between a command and speech. Participants usually move the tongue in the sagittal plane of the mouth while speaking unlike assigning a command. The sagittal plane tongue position is trained as a resting command of the classification algorithm. Landmarks other than the resting (LEFT, RIGHT, UP, DOWN, LS, RS) are chosen in a way so that the tongue must move away from the sagittal plane to assign a command. By incorporating these techniques, the current algorithm can differentiate between a tongue command and speech. This phase of development is benchmarked and compared with the existing assistive technologies in Table 1

2.3 System Architecture: Phase III

The mTDS wearable unit is divided into two parts: 1) a sensor unit where the users' remaining abilities are captured, and 2) a control unit where users' intentions are detected by processing the captured data. Figure 9 illustrates the data flow to control a device, such as PC or smartphone. Six discrete tongue commands are converted into mouse clicks (left/right select: LS, RS) and the equivalent of keyboard four arrow keys. Head motion is used as a proportional control input, where pitch and roll angles are utilized for cursor navigation in two dimensions, in any direction, at any speed within a range. Yaw is not used to prevent unintentional head movements, such as looking around, to be confused as user commands. Rapid typing and dictation are also supported by automatic speech recognition (ASR) software, running on the host platform. A more detailed description of the hardware is provided in the following.

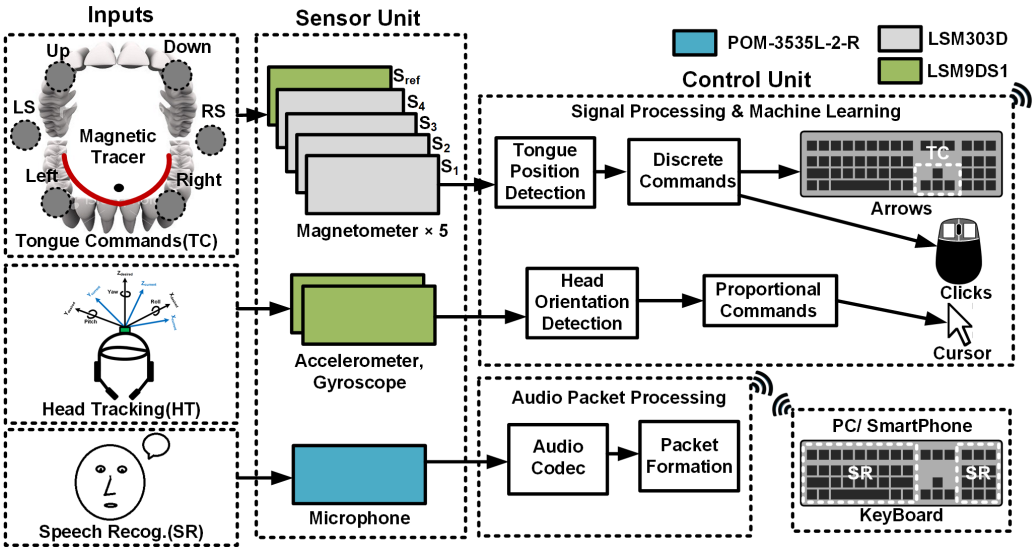

Figure 9: The main architecture of the final mTDS which can capture and process the gestures in system.

2.3.1 Sensor Unit

Four 3-axial magnetometers (LSM303D, STMicroelectronics) track tongue movement with an additional magnetometer used as a reference to attenuate magnetic interference in real time, including the earth's magnetic field (EMF). The reference magnetometer is mounted

in the Control Unit, >10 cm away from the small magnetic tracer attached to the tongue to only capture the EMF. It is combined with a 3-axial accelerometer and a 3-axial gyroscope in a 9-D inertial measurement unit (IMU) (LSM9DS1, STMicroelectronics), which is used to measure the pitch and roll of the user's head. A small microphone (POM-3535L-2-R) is mounted on the left arm of the mTDS headset near the user's mouth to capture speech, as shown in Figure 10.

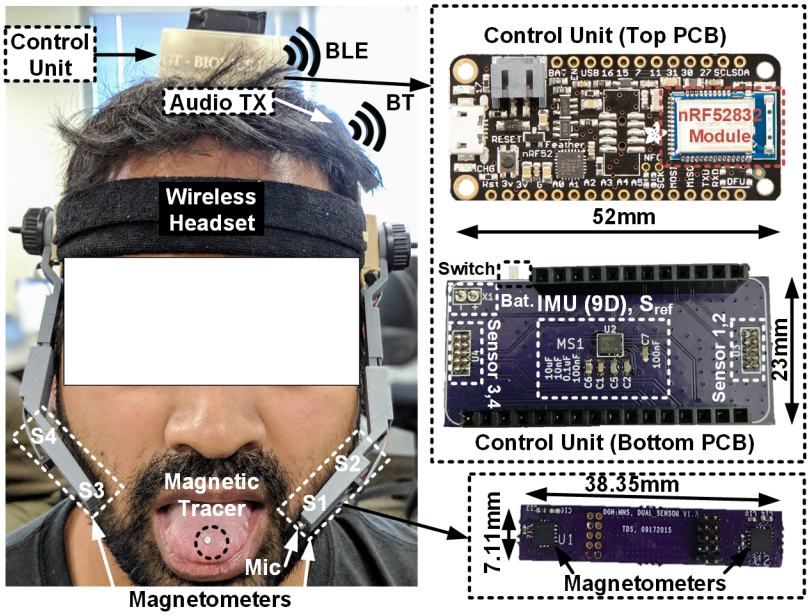

Figure 10: The wearable headset captures 3 different inputs (tongue commands, head movement and speech), processed the gestures and seamlessly interface with devices using Bluetooth

2.3.2 Control Unit

The Control Unit is composed of three sub-units for 1) voice recording, 2) processing, and 3) tongue/head motion tracking. The first sub-unit digitizes the audio signal captured by the microphone on the left arm of the headset, and transmits as digitized raw audio data to a PC/smartphone via a commercial-off-the-shelf (COTS) Bluetooth transceiver (Jabra, Denmark). The second sub-unit is based on a COTS MCU platform, called Bluefruit (Adafruit, New York, NY), shown in Figure 10, which is based on an advanced low-power MCU, nRF52832 (Nordic Semiconductor, Norway). This module enables efficient and real-time

processing of tongue and head gestures by an ARM Cortex M4 core with built-in hardware-based floating-point unit (FPU). An interface to all 3-axial magnetometers on the poles and the IMU is provided by a custom-designed PCB cape for the Bluefruit platform, as shown in Figure 10, which also includes serial peripheral interface (SPI) ports, a power switch, and a battery connector. LED indicators for system status and a micro-USB port to recharge the 700 mAh lithium-polymer (Li-Po) battery are all embedded in the control unit 3D-printed enclosure on top of the headset. Thanks to the mTDS onboard processing power, user inputs are directly translated into discrete-&-proportional commands (accessible via tongue-&-head voluntary movements), just like keyboard-&-mouse (KnM) combination, and transmitted via BLE to target devices, PC/smartphone in this case, using the HID protocol.

2.3.3 Processing Unit

A wheelchair driving interface is needed to communicate with a headset to receive wireless data for calibration and training. Once the calibration and training are done, it can be used to drive a wheelchair using different modes. This interface will have a touchscreen display to receive user inputs and provide visual feedbacks during driving. The interface electronics of the wheelchair generates two analog voltages, referred to as vectors, to go forward/backward and turning left/ right. It can generate different levels of voltages to increase/decrease the vector amplitudes. It will have an emergency stop functionality to stop the wheelchair in case of lost connection or unexpected magnetic interferences to the magnetometers. This interface will be controlled by the BBB as discussed in previous phase of the development.

2.3.4 Touchscreen User Interface

The purpose of touchscreen interfaces is to receive user inputs during the calibration, training and provide options to the user to choose different modes to drive the wheelchair by following different strategies. The user interfaces to check the waveform, calibrate, train, test and drive the wheelchair is shown in Figure 11.

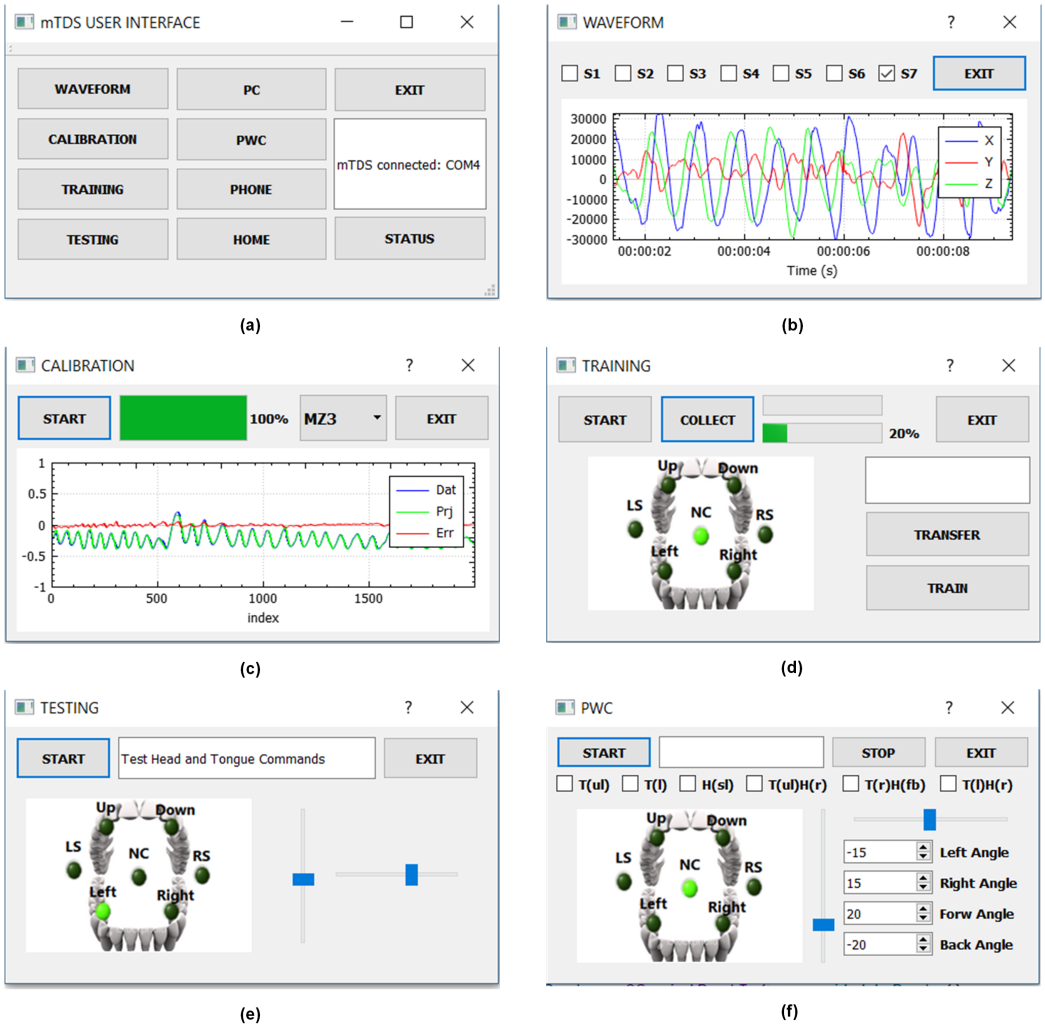

Figure 11: The touchscreen user interface (UI) which is implemented in a single board computer BeagleBone Black with touchscreen display (a) Main UI navigates users through different windows (b) waveform UI to check the functionality of the system which shows the real-time status of all magnetometers, accelerometer, and gyroscope (c) Calibration UI to calibrate and generate metrices to attenuate earth magnetic field (EMF) (d) Training UI to train 7 tongue commands (e) Test UI to test the quality of training before driving the wheelchair (f) Wheelchair driving UI to provide visual feedback as well as configure thresholds of head motions and select different driving modes to evaluate.

CHAPTER 3

GESTURE RECOGNITION AND ALGORITHMS

3.1 Tongue Gesture Processing

The mTDS has shown significant improvements regarding performance, user experience can be improved further by applying advanced signal processing techniques in generating tongue commands with better accuracy. Previously, to train tongue commands, it was required $\sim 12mins$ ($10\times$ repetition of each command, total seven commands each $10s$) of training session which made the user tired at the beginning of the usage. Bad training data were removed manually from a 3D principal component analysis (PCA) plot which resulted in erroneous classification model for evaluation. A small shift of the wearable headset required retraining of the tongue commands. Since mTDS uses the speech, tongue, and head movements, it is necessary for the tongue gesture processing algorithm to be more robust when the head movement is triggered with tongue motion. The algorithm also needs to differentiate between the speech, sneezing, swallowing, coughing, eating, drinking and tongue command without requiring any additional sensor inputs. Previously, tongue command processing required additional smartphone/ PC / processing unit which makes the system more dependable on those devices. An optimization is needed to process both the tongue and head movements in the wearable unit and seamlessly interface with multiple devices just by sending a Bluetooth human interface device (HID) commands.

The mTDS currently has five different magnetometers (four around the cheek, one as a reference located on top of the headset) to find the magnetic tracer location which is attached to the tip of the tongue. There are many approaches to track the attached magnetic tracer. The traditional way is to solve magnetic dipole equation shown in Equation (1) [53].

$$\vec{B}(\vec{R},\vec{M}) = \frac{\mu_0}{4\pi}\frac{3(\vec{M}.\vec{R})\vec{R} - \|\vec{R}\|^2\vec{M}}{\|\vec{R}\|^5} \tag{1}$$

where $\vec{B}$ is magnetic flux density generated by the magnetic tracer, $\vec{M}$ dipole moment

generated by the tracer. $\vec{R} = \vec{s} - \vec{a}$, $\vec{s}$ is the location of the sensor and $\vec{a}$ location of the magnetic tracer. The magnetic dipole moment, $\vec{M}$, depends on the orientation of the tracer. The position of the magnetic tracer $\vec{a}$ can be found by solving the Equation (1) for four magnetometers, but this is a nonlinear optimization problem that requires a lot of computation which is not feasible to execute in a wearable embedded system.

Another approach of finding the magnetic tracer location is to solve the triangulation problem using four magnetometers [54]. Since the relative location of the sensors varies due to the head movements and participants facial shape, this method is not suitable for this application. However, The main goal is not to find the exact location of the magnetic tracer but to find the current tongue gesture (approximate position) where the magnet is located which is defined as a command. In this work, we approach this problem using different machine learning techniques such as k-Nearest Neighbour (KNN), Logistic Regression (LR), and Support Vector Machine (SVM) with different kernels (linear, radial basis function). In order to process the tongue commands, it is necessary to attenuate all other magnetic field contributed by unexpected sources such as Earth magnet, soft and hard iron. Previously, a projection and subtraction method was used to attenuate the interference where the reference magnetometers are also affected by the tongue magnetic tracer [51]. In order to improve the quality of noise attenuation, a hardware upgrade is made, and a reference magnetometer is added to the control unit to collect dynamic magnetic field interferences where Earth magnetic field (EMF) is the major contributor.

Previously, human data was utilized to test the performance of the implemented algorithm. However, the feasibility of repeating the same test to develop a new algorithm is not reproducible in case of hardware changes, for example, adding additional magnetometer. The outcome of the test is also biased on the experience of participants to TDS [51]. An experienced participant is more likely to assign the same command precisely in the same way during training and testing, unlike a naive participant. Therefore, a new approach is introduced by emulating the tongue movement using a four-dimensional robot. Using the proposed method, the magnet can be moved to any position in three-dimensional space as well as orientation (θ) can be changed at any angle. Finally, based on the performance of the

different algorithms, we proposed an optimized algorithm, and test it with 15 able-bodied participants to verify the functionality.

3.1.1 Methods

The precise magnet position (ground truth) is difficult to find without complex instrumentation. The performance of the tongue command processing algorithm is also subjective to the user's expertise with the mTDS. Therefore, two evaluation studies were conducted, with precise ground truth using a 4D robot and with human participants. For both evaluations, the same mTDS headset was used to collect the magnetometers data and saved offline to find the performance.

3.1.2 Robot Data Collection Protocol

An experimental setup is developed to emulate human tongue movement dynamics using a precise ($0.1\mu m$) four-dimensional robot shown in Figure 12. Seven different locations are chosen to emulate tongue commands including a resting position. Since it is not possible for a user to place the magnetic tracer precisely at the same location and orientation, magnet location and orientation for each command was varied to consider that fact. The robot consists of four stepper motors. Three of them control the x, y, and z position of the magnetic tracer to move it to a specific location. The fourth stepper motor is used to control the magnetic tracer orientation. A wooden platform is designed to place the mTDS headset in front of the robot. An application, TinyG, is used to send commands from the computer to control stepper motors. A custom-designed user interface software was used to send the position and orientation (x, y, z, θ) commands to TinyG and collect magnetometer data from the mTDS headset using USB communication. After sending the commands, magnetic tracer moved to the desired location and orientation then the custom software collected the data from five magnetometers.

Tongue command locations shown in Figure 13 are chosen based on the human tongue and mouth anatomy. Each command location is varied with an offset of $2mm, -2mm, 5mm$, and $-5mm$ in x, y, z direction. The orientation of the magnetic tracer in each location was also varied by changing the θ to $5^\circ, -5^\circ, 10^\circ$, and -10°. The data collections were repeated

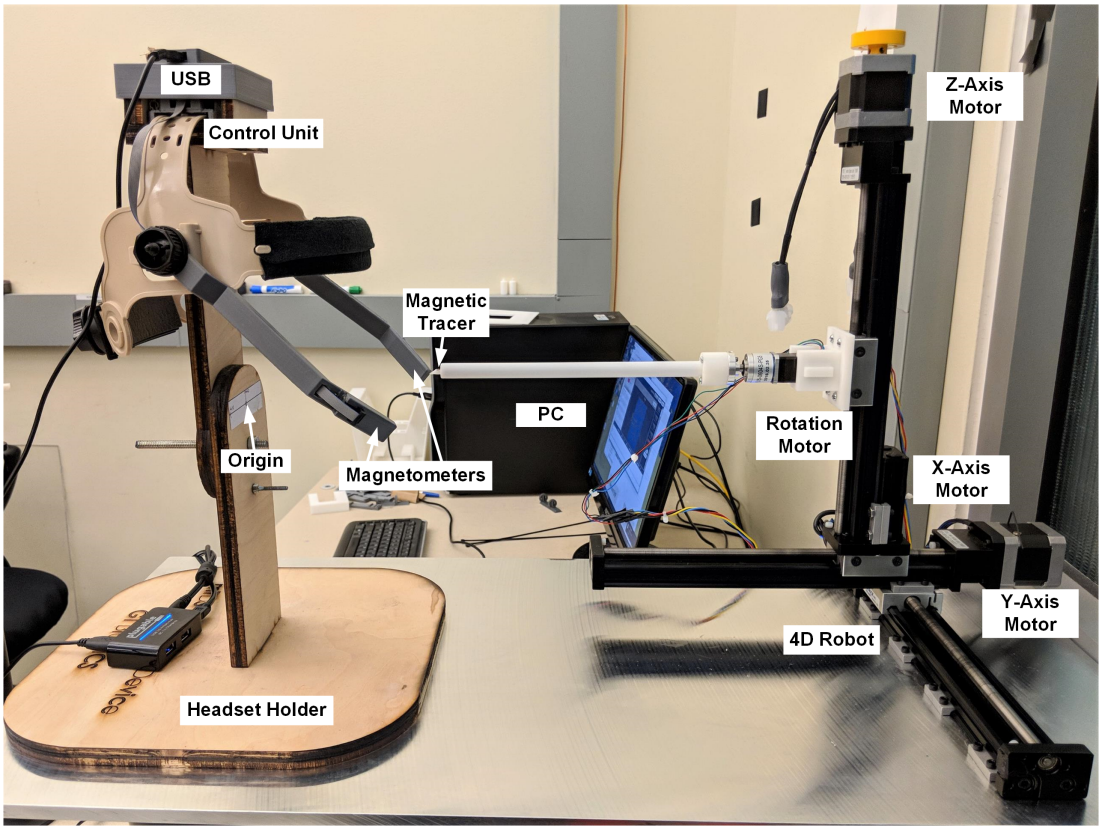

Figure 12: Experimental setup for 4-dimensional robot to move the magnet tracer to collect data. Here, 3 motors move the magnet tracer to the seven landmarks defined before, rotation motor rotate the magnetic tracer in θ direction.

three times for each case. The sampling frequency of the magnetometers was $100Hz$. The data were captured for $1s$ thus 100 samples in total 300 samples with three repetitions for each command location and orientation.

3.1.3 Human Data Collection Protocol

Fifteen able-bodied volunteers (C01-C15), eight males and seven females between the ages of 18 to 35 years old, one of whom was experienced, one was intermediate, and 13 naive with respect to mTDS, participated in this study. The protocol was approved by the institutional review board (IRB) at the Georgia Institute of Technology. The participant's tongue was dried with tissue and a small disk-shaped magnetic tracer ($\phi 4.8mm \times 1.5mm$, K&J Magnetics, Jamison, PA), sterilized using isopropyl alcohol, was attached to the tongue, 1 cm from the tip using a cyanoacrylate tissue adhesive.

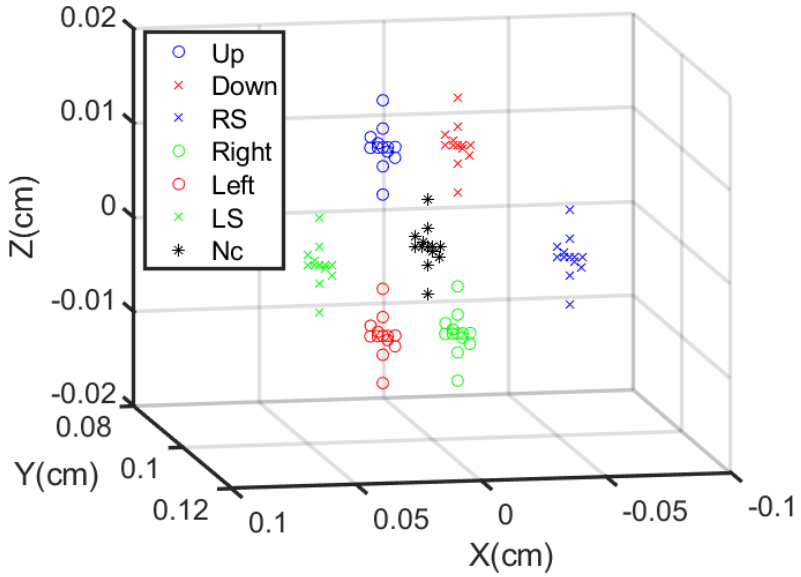

Figure 13: Data points captured for testing different algorithms using seven different commands including a neutral.

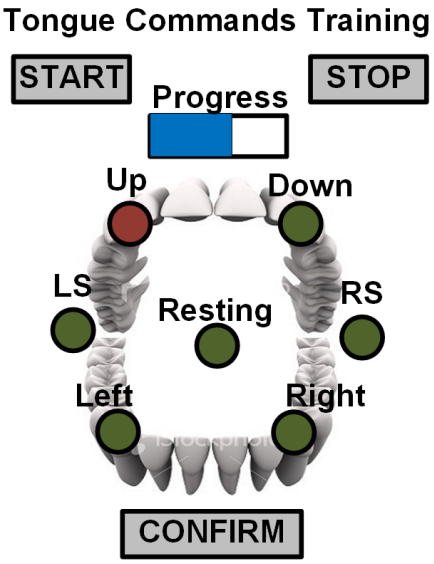

Figure 14: User interface to collect 7 tongue commands. start and stop buttons were used to start and end the training process. Confirm button was pressed when participants place the tongue to predefined landmarks. A progress bar showed the data capturing progress.

Before the tongue command data capture for training/ testing, calibration data was collected by placing user's tongue at resting position and move the head around for 20 s. A user interface (UI) shown in Figure 14 was used to instruct participants to place the magnetic tracer to the shown landmarks to collect the magnetometer data. Participants

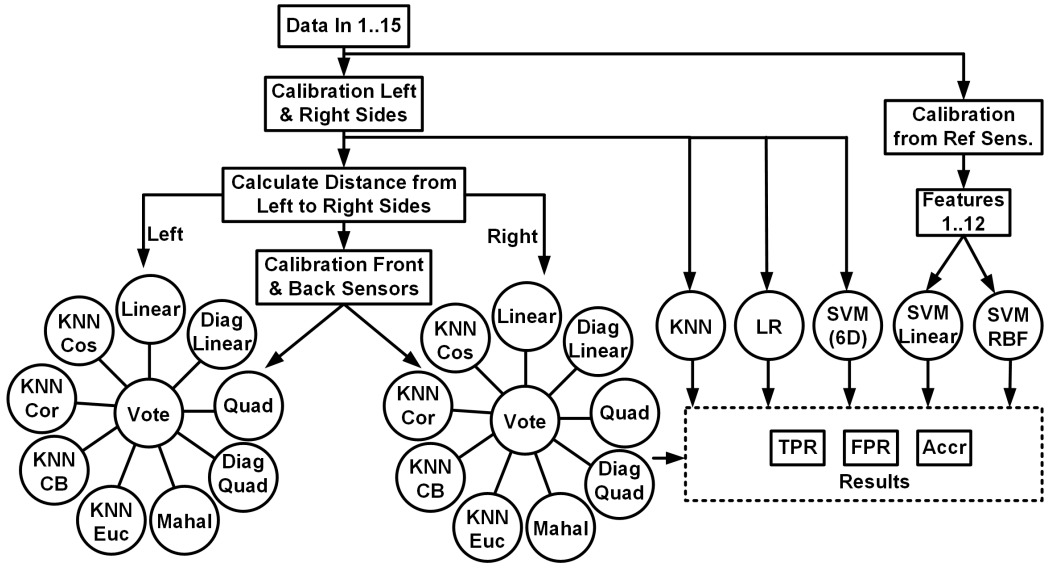

Figure 15: Six different algorithms are used to find the performance of tongue command detection the classification accuracy.

kept the tongue to the same location until the data collection was completed. The data collection progress was shown using a status bar in the UI. Each command was captured for $10s$ total 1000 samples. Each round of data collection consisted of seven tongue commands (Left, Right, Up, Down, LS, RS) including a resting (Nc) position. Total four rounds of data were collected from each participant to be analyzed later.

3.1.4 ALGORITHMS

Six different algorithms are tested and compared with the proposed algorithm. Some of the algorithms were used in the previous work [51, 55].Figure 15 presents a flowchart of six different algorithms.

3.1.4.1 k Nearest Neighbor with 9 distances (KNN-9)

This algorithm was used in the previous version of TDS [23, 51]. KNN-9 divides tongue movements into two different sets of commands as shown in Figure 15 after attenuating EMF interferences using a projection and subtraction method [51]. Left block commands are responsible for processing Left, Up, LS, and Resting. Right block processes Right, Down, RS, and Resting commands. EMF attenuated feature vector is fed to 9 different linear and nonlinear classifiers (linear, diagonal linear, quadratic, diagonal quadratic, Mahalanobis

minimum distance, and four KNN classifiers). A majority voting schema is used to find the final result.

3.1.4.2 k Nearest Neighbor (KNN)

KNN is a simpler version of KNN-9 and a potential option to be implemented in the mTDS. 3-Nearest Neighbour classifier is used to train the tongue commands after attenuating the EMF interferences using projection and subtraction technique [51]. The feature vector after attenuating the interferences is fed into the algorithm to train and test the classification performance. KNN only uses one distance instead of 9 (previous version) thus does not need any voting schema to find the assigned command.

3.1.4.3 Logistic Regression (LR)

This algorithm was proposed in past by [55] to be implemented in a standalone TDS. Logistic Regression classifier is used after attenuating the EMF interference using a projection and subtraction method mentioned in [55]. One vs. all classifier is used to find the assigned tongue command.

3.1.4.4 Support Vector Machine Linear kernel (SVM-6D)

SVM-6D is implemented to find the effect of the added reference magnetometer in the CU to attenuate EMF. It utilizes only four magnetometers' data to find the user tongue commands. Two magnetometers each side of the cheek (M2, M4 in Figure 10) are used as the references to capture dynamic EMF interferences. During the training and testing, it projects the reference magnetometer (M2 to M1 and M4 to M3) to other sensors and subtracts to find the six-dimensional input feature. Input feature vector is fed to 21 one vs. one classifier to find the results. A maximum vote win schema is implemented to find the assigned tongue command from the results.

3.1.4.5 Support Vector Machine RBF kernel (SVM- RBF)

This algorithm compares the effect of changing the kernal function of classifier to the performance.This algorithm incorporates an additional reference magnetometer located in the CU of the mTDS headset to attenuate EMF interferences. After attenuating the noise, 12D

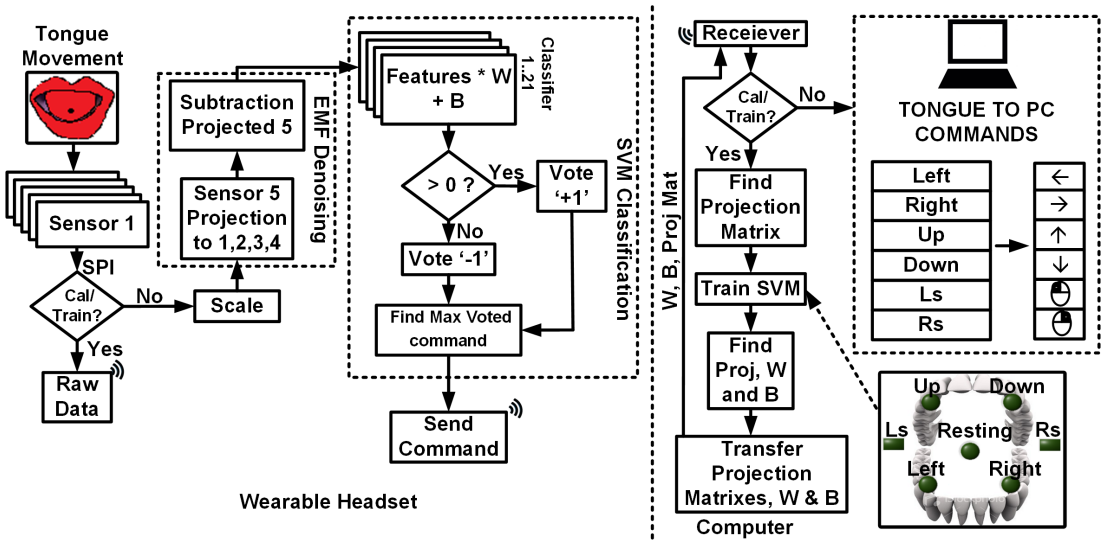

Figure 16: Support vector machine with linear kernel is used to implement in the final hardware. The firmware architecture for the wearable headset (left) is presented here. Wearable headset plays two different roles while training and testing. During training, it sends raw magnetic sensor data to PC to generate projection matrices, W and B which are used during the testing of the system. It sends only tongue commands which converts to inputs (right) to control a PC in real time during the evaluation period.

input feature vector is used to train and test the algorithm performance using support vector machine with radial basis function (RBF) kernel.

3.1.4.6 Support Vector Machine linear kernel (SVM-Linear)

12 features are extracted after attenuating EMF using the reference magnetometer on the CU. These 12 features are used to train and test the algorithm accuracy. LIBSVM [56] is used to train the seven tongue commands using one vs. one support vector machine with a linear kernel. The flow diagram of the algorithm processing during training and execution is shown in Figure 16. The proposed algorithm is implemented in the embedded system using two state machines. The first state machine transfers the five magnetometers data to a computer to calibrate and train. Once training is done, model files and projection matrices are transferred back to the mTDS headset to save it in the nonvolatile memory of the microcontroller unit (MCU). Second state machine utilizes the calibration and training matrices to attenuate interference and generates input features to find the classification results using the following steps.

The first step of the algorithm requires scaling the magnetometer data. A factor of 10 is used to scale the data using Equation (2).

$$\begin{bmatrix} M_x & M_y & M_z \end{bmatrix}_{1..4,r} = \frac{\begin{bmatrix} M_x & M_y & M_z \end{bmatrix}_{1..4,r} * 10}{2^{15}} \tag{2}$$

where M_x, M_y, and M_z are the x, y, and z axis magnetometer data, respectively. $1, 2, 3, 4$, and r represents M_1, M_2, M_3, M_4, and S_{ref} magnetometer data, respectively. After scaling, calibration (20 s) data are used to estimate the gains and offsets of the four projection matrices that project the reference magnetometer's axes to each of the four magnetometer of the mTDS headset, using the least square error (LSE) method shown in Equation (3),

$$\begin{bmatrix} o_x & o_y & o_z \\ g_{xx} & g_{yx} & g_{zx} \\ g_{yx} & g_{yy} & g_{yx} \\ g_{zx} & g_{zy} & g_{zz} \end{bmatrix}_{1..4} = \begin{bmatrix} 1 & M_{x,r} & M_{y,r} & M_{z,r} \end{bmatrix}^{-1} \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix}^T_{1..4} \tag{3}$$

where o_x, o_y, o_z are the offsets, g_{xx}, g_{yy}, g_{zz} are the gains, and $g_{xy}, g_{xz}, g_{yx}, g_{yz}, g_{zx}, g_{zy}$ are the cross gains, while $M_{x..z,r}$ and $M_{x,y,z}$ represent the reference and headset arms magnetometers' measurements, respectively.

Training data ($7 \times 3 \times 10s$) are first scaled, then calibrated using Equation (4), Equation (5) to generate the 12-D input features.

$$\begin{bmatrix} Mprj_x \\ Mprj_y \\ Mprj_z \end{bmatrix}_{1..4} = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix}^T_{1..4} \begin{bmatrix} g_{xx} & g_{yx} & g_{zx} \\ g_{yx} & g_{yy} & g_{yx} \\ g_{zx} & g_{zy} & g_{zz} \end{bmatrix}_{1..4} + \begin{bmatrix} o_x \\ o_y \\ o_z \end{bmatrix}_{1..4} \tag{4}$$

$$\begin{bmatrix} feature_x \\ feature_y \\ feature_z \end{bmatrix}_{1..4} = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix}_{1..4} - \begin{bmatrix} Mprj_x \\ Mprj_y \\ Mprj_z \end{bmatrix}_{1..4} \tag{5}$$

where, $Mprj_{x,y,z}$ are the projected 3-axial reference sensor values and $feature_{x,y,z}$ are 3-axial features generated from the four magnetometers. These features are used to train tongue commands which generated weight ($W_{21\times12}$) and bias ($B_{21\times1}$) matrixes of the 21

(one vs. one, seven commands) linear binary support vector machine classifiers, using the LIBSVM library [56]. During the execution,magnetometer data are processed following the same steps using Equation (4), Equation (5), respectively. 12-D input features are classified using Equation (6) and the maximum voted command, y is derived from the results, using Equation (7).

$$res_{21\times1} = W_{21\times12} feature_{12\times1} - B_{21\times1} \tag{6}$$

$$y = \underset{n=0..6}{\text{Max}}\ vote(res_{21\times1}) \tag{7}$$

where, $res_{21\times1}$ is the 21 classification results.

To optimize the computation during execution, Equation (2) is changed to Equation (8). This reduces 15 multiplications during each command processing.

$$\begin{bmatrix} M_x & M_y & M_z \end{bmatrix}_{1..4,r} = \frac{\begin{bmatrix} M_x & M_y & M_z \end{bmatrix}_{1..4,r}}{3676.8} \tag{8}$$

The calibration Equation (4) is changed to Equation (9), $B_{21\times1}$ is modified to $Bm_{21\times1}$ using Equation (10) and Equation (11) is used instead of Equation (6) to find $res_{21\times1}$. Since $Bm_{21\times1}$ is computed offline during the training step, it requires to calculate 12 less additions during each command execution.

$$\begin{bmatrix} Mprj_x \\ Mprj_y \\ Mprj_z \end{bmatrix}_{1..4} = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix}_r^T \begin{bmatrix} g_{xx} & g_{yx} & g_{zx} \\ g_{yx} & g_{yy} & g_{yx} \\ g_{zx} & g_{zy} & g_{zz} \end{bmatrix}_{1..4} \tag{9}$$

$$Bm_{21\times1} = W_{21\times12} \begin{bmatrix} o_x \\ o_y \\ o_z \end{bmatrix} + B_{21\times1} \tag{10}$$

$$res_{21\times1} = W_{21\times12} feature_{21\times1} - Bm_{21\times1} \tag{11}$$

Table 2: Outcome of the algorithms by varying the position of the magnetic tracer from each landmark.

Algorithm	Accuracy (%)	Sensitivity (%)	Specificity (%)
KNN-9	20.8	18.2	100
KNN	100	100	100
LR	96.3	100	99.3
SVM(RBF)	89.6	100	99.1
SVM(6D)	99.6	100	100
SVM(Linear)	100	100	100

3.1.5 Results

Tongue emulated data using four-dimensional robot are analyzed into three different categories by varying magnetic tracer positions, orientation, and the combination of both. Finally, all the algorithms are tested by the human participants' data to validate the outcome in real application.

3.1.5.1 Robot Data: Position Variation

During this analysis, the magnetic tracer orientation was fixed to 0 deg. Three different position data are used to train the algorithms and rest to test the performance. Only three variations of the data are used because we want to limit the training data collection to three repetitions to reduce user fatigue at the beginning of the mTDS usage. Total 20 different variations of positions cases (5 different positions, 3 for training, 2 for testing) are used to train and evaluate the performance. The overall performances are reported in terms of accuracy, sensitivity, and specificity in Table 2.

KNN-9 accuracy is 20.8% which is significantly lower than other algorithms. KNN and SVM (Linear) accuracies are the highest (100%) for all the magnet position variations. However, LR and SVM(RBF) accuracies are 96.3% and 89.6%, respectively which are slightly lower than SVM(Linear) and KNN. SVM (6D) accuracy is 99.6% which is similar to SVM (Linear).

Table 3: Outcome of the algorithms by varying the orientation of the magnetic tracer.

Algorithm	Accuracy (%)	Sensitivity (%)	Specificity (%)
KNN-9	21.6	25.3	100
KNN	100	99.9	100
LR	84.7	77.8	96.3
SVM(RBF)	98.9	96.6	100
SVM(6D)	99.8	99.8	100
SVM(Linear)	100	100	100

3.1.5.2 Robot Data: Orientation Variation

In this case, the position of the magnetic tracer was fixed to the same locations for each command. But the orientation (θ) is changed to train and test the algorithms. Three different orientation data are used to train the algorithms and rest to test the performance. Total 20 different orientation cases (5 different orientations, 3 for training, 2 for testing) are evaluated. The overall performances are reported in terms of accuracy, sensitivity, and specificity in Table 3.

The performances due to the orientation variation results in the similar outcome (accuracy: 100%) for both KNN and SVM (Linear). However, in this case, KNN-9 and SVM (RBF) performances are better than the magnet position variation. KNN-9 accuracy and sensitivity are improved by 0.8% and 7.1%, respectively. SVM (RBF) accuracy is improved by 9.3% but the sensitivity decreased by 3.4%. LR accuracy, sensitivity, and specificity are decreased by 11.6%, 22.2%, 3% due to magnet orientation variation compared to orientation, respectively.

3.1.5.3 Robot Data: Position & Orientation Variation

Both the magnetic tracer position and orientation were varied to evaluate the performances of each algorithm. Three different orientation and position data are used to train and rest to test the algorithm performances. Total 160 different cases (5 different positions, 5 different orientations, a combination of 3 for training, rest for testing) are used to evaluate the performances. The overall performances are reported in terms of accuracy, sensitivity, and specificity in Table 4.

Variation of both orientation and position results in better sensitivity (37.9%) using

Table 4: Outcome of the algorithms by varying both the orientation and position of the magnetic tracer.

Algorithm	Accuracy (%)	Sensitivity (%)	Specificity (%)
KNN-9	19.2	37.9	99.2
KNN	97.7	96.1	99.2
LR	86.6	87.5	95.8
SVM(RBF)	70.4	88.6	90.1
SVM(6D)	95.3	89.9	98.4
SVM(Linear)	98.9	98.4	99.3

KNN-9 with less accuracy (19.2%). KNN results in 2.3%, 3.8%, and 0.8% reduced accuracy, sensitivity, and specificity compared to magnet orientation and position variation, respectively. Using LR, the accuracy and sensitivity are improved by 1.9% and 9.7%, respectively compared to orientation variation. However, specificity is reduced by 0.5%. Both magnet position and orientation variation result in lower accuracy, sensitivity, and specificity using LR compared to only position variation. The accuracy, sensitivity, and specificity using SVM (RBF) algorithm are 28.5%, 8%, and 9.9% less respectively compared to only magnet orientation variation. Accuracy, sensitivity, and specificity are 19.2%, 11.4%, and 9% less compared to magnet position variation. Using SVM(6D), accuracy, sensitivity, and specificity are 4.5%, 9.9%, 1.6% and 4.3%, 10.1%, 1.6% less compared to magnet orientation and position variation respectively. Accuracy, sensitivity, and specificity are reduced by 1.1%, 1.6%, 0.7% compared to magnet orientation and position variation using SVM (Linear) algorithm. It can be observed that among all algorithms SVM (Linear) shows the least variation in performance due to the change of magnet position, orientation or both. The performance of SVM(Linear) is also better compared to other options in all situations as well.

3.1.5.4 Evaluation by Human Data

Four-fold validation technique is used to find the classification outcome. Each time three rounds of training data are used to train the classifier and rest one for testing. All results (15 participants, four rounds) are accumulated to report the overall accuracy, sensitivity, and specificity of the algorithms in Table 5. The overall accuracy and specificity using SVM(Linear) and KNN are 96.1% and 99.3%, respectively which are higher than the other

Table 5: Outcome of the algorithms from 15 participants using four fold validation technique.

Algorithm	Accuracy (%)	Sensitivity (%)	Specificity (%)
KNN-9	67.1	54.6	94.7
KNN	96.1	93.9	99.3
LR	91.4	88.3	99.4
SVM(RBF)	64.1	64.1	91.6
SVM(6D)	95.1	94.6	98.9
SVM(Linear)	96.1	94.7	99.3

Table 6: Confusion table between tongue command(TC) vs. no command(NC).

Algorithm	TP (%)	TN (%)	FN(%)	FP(%)
KNN-9	69	12.3	16.7	2.0
KNN	84.9	14	0.8	0.3
LR	83.8	14.2	1.9	0.3
SVM(RBF)	77.3	8.1	8.4	6.2
SVM(6D)	84.2	14.2	1.5	0.1
SVM(Linear)	84.7	14.2	1.0	0.1

algorithms. The sensitivity of SVM (Linear) is 94.7% which is highest among other options.

Finally, It is important to choose an algorithm which is more conservative regarding assigning tongue commands. It is always preferable to assign a no command (Nc) / resting instead of a false positive tongue command in a confused situation.Table 6 shows the confusion matrices between the tongue commands (TC) and no command (Nc) / resting. It can be observed that the SVM (Linear) algorithm is more conservative (0.1% false positive) among others regarding assigning tongue commands. In this analysis, the total true positives and true negatives are 85.7%(359640) and 14.3%(59940), respectively.

3.2 Head Movement Processing

The Proportional Head Control routine gives a user the ability to move a mouse cursor in any direction and at any speed by tilting their head. This capability can deliver the usability of modern computer mice to individuals with disabilities. The routine is written in MATLAB (Mathworks Inc, Natick, MA USA) and integrated into LabView for real-time operation using MathScriptRT blocks. The algorithm relies on the LSM9DS1 for sensory input.Section 3.2 depicts a functional block diagram of the proportional head control

algorithm. The blocks are grouped into three categories.

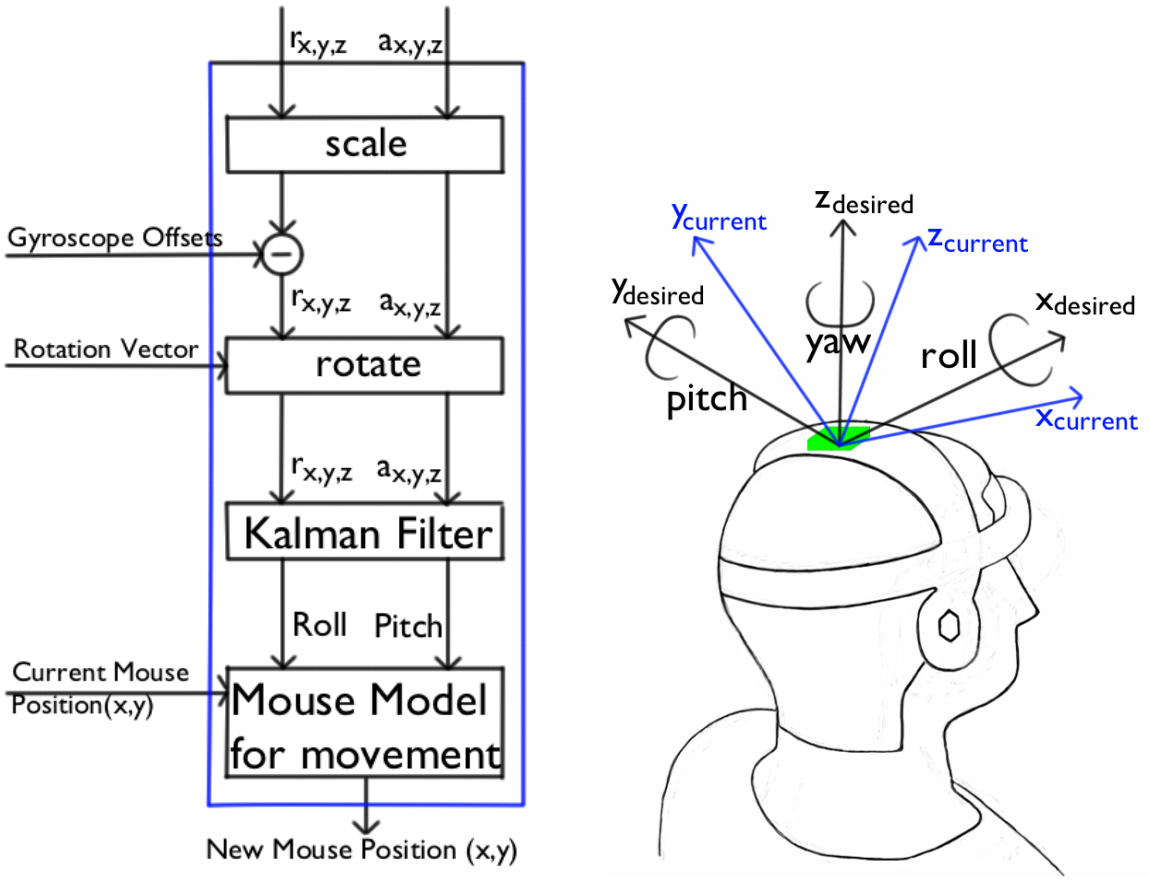

Figure 17: (a). Functional blocks comprising the Proportional Head Control routine (within blue box) and its I/O interactions with the mTDS LabVIEW application. The blocks illustrate the sequence in which they transform the sensor data into mouse movement. (b). LSM9DS1 reference axes (blue) may differ from physical axes (black) for the user based on how the MPU (green box) is mounted on the mTDS chassis.

3.2.1 Calibrate Sensor Data

The mTDS tracks Pitch and Roll only. Magnetometer data from the LSM9DS1, needed for calculating Yaw is unused. Calibrating 6 axes of accelerometer ($a_{x,y,z}$) and gyroscope ($r_{x,y,z}$) sensor readings from the LSM9DS1 involve three transformations.

Firstly, the sensor readings reported as 16 bit unsigned integers [0-65535] at 100 Hz need to be scaled to the dynamic ranges of the LSM9DS1 accelerometer ($\pm 2g$) and gyroscope ($\pm 250°/second$) per Equation (12) and Equation (13).

$$a_{x,y,z} = \frac{Accelerometer\ Range \times (a_{x,y,z} - Offset)}{Scale\ Factor} \tag{12}$$

$$r_{x,y,z} = \frac{Gyroscope\ Range \times (r_{x,y,z} - Offset)}{Scale\ Factor} \tag{13}$$

The $Offset$ value of 32768 represents the zero value for sensor readings. The $Scale\ Factor$ of 32768 relates the maximum magnitudes of sensor readings to the physical values they represent ($AccelerometerRange = \pm 2g, GyrosocopeRange = 250°/second$).

Secondly, the gyroscope offsets are subtracted from the scaled $r_{x,y,z}$ sensor readings from (13). MEMS gyroscopes like ones used by the MPU 9250 rely on the Coriolis effect to detect angular rotation. At rest, their readings include DC and low frequency noise that is difficult to distinguish from actual rotation. Combating this noise is essential to prevent accumulation of significant "Gyroscope Drift" errors in pitch/roll calculations during continuous operation [57].

A one-time calibration procedure determines the DC noise. This is achieved by averaging ∼1 minute of gyroscope data from the LSM9DS1 when the headset is at rest and in an upright position.

$$Gyroscope\ Offset\ r_{x,y,z} = \sum_{t=0}^{t=60sec} \frac{r_{x,y,z}[t]}{92\ Hz \times 60\ sec} \tag{14}$$

During normal operation, each reading is corrected as follows.

$$r_{x,y,z} = r_{x,y,z} - Gyroscope\ Offset\ r_{x,y,z} \tag{15}$$

Finally, the accelerometer and gyroscope sensor readings must be rotated to the upright position aligned with the user's reference axes as depicted in Section 3.2. A vector which can apply this rotation Equation (18) to sensor readings is calculated by keeping the headset stationary in an upright position with a known/desired acceleration vector A_d Equation (16) and determining the current orientation A_c of the LSM9DS1 Equation (17) by averaging accelerometer data captured during the MPU 9250 gyroscope offset calibration procedure described in the previous step.

$$A_d = 0\hat{i} + 0\hat{j} - 1\hat{k} \tag{16}$$

$$A_c = A_x\hat{i} + A_y\hat{j} + A_z\hat{k} \tag{17}$$

The rotation vector is then determined as follows.

$$GG = \begin{bmatrix} A_c \cdot A_c & -|A_c \times A_d| & 0 \\ |A_c \times A_d| & A_c \cdot A_d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$FF = \begin{bmatrix} A_c & \frac{A_d - A_c \cdot A_d \times A_c}{|A_d - A_c \cdot A_d \times A_c|} & A_d \times A_c \end{bmatrix}$$

$$Rotation\ Vector = FF \times GG \times FF^{-1} \tag{18}$$

During normal operation, the rotation vector is applied as follows.

$$a_{x,y,z} = Rotation\ Vector \times a_{x,y,z} \tag{19}$$

$$r_{x,y,z} = Rotation\ Vector \times r_{x,y,z} \tag{20}$$

3.2.2 Deduce Head Orientation

Pitch and Roll control mouse movement. They can be determined either using accelerometer reading only Equation (21), Equation (22) or integrating gyroscope readings over time.

$$Pitch = \tan^{-1}(\frac{-a_x}{a_z}) \times \frac{180^\circ}{\pi} \tag{21}$$

$$Roll = \tan^{-1}(\frac{a_y}{a_x^2 + a_z^2}) \times \frac{180^\circ}{\pi} \tag{22}$$

Both methods have limitations. Accelerometer readings inherently include high frequency noise and are also vulnerable to sudden jerky movements. Gyroscope readings are affected by "Gyroscope Drift" [57] mentioned earlier. Kalman filters are an attractive method of fusing 2 signals with different levels of noise together to achieve a resultant signal which has lesser noise than either of the component signals [58]. Commonly used in inertial navigation, they are an appropriate choice here to fuse accelerometer and gyroscope sensor readings and determine robust Pitch and Roll signals for tracking head orientation.Equation (23) and Equation (24) describe the state model.

$$x_t = A \times x_{t-1} + w_{t-1} \tag{23}$$

$$z_t = H \times x_t + v_t \tag{24}$$

$$\text{Where } A = \begin{bmatrix} 1 & \Delta\, t \\ 0 & 1 \end{bmatrix}, \; x = \begin{bmatrix} Pitch & Roll \\ r_y & r_x \end{bmatrix}, \text{H is a } 2 \times 2$$

Identity matrix, Δt is the sampling period and w and v represent the process and measurement noise, respectively.

3.2.3 Translate Head Orientation to Mouse Movement

The mouse movement model converts angular displacement of the head along the Roll and Pitch axes to mouse cursor velocities along the X and Y axes. Residual head movement when the user intends to stay in resting position can result in small undesired movement of the mouse cursor. To provide robustness against residual movement, a threshold function ($\lambda_{roll,pitch}$) is incorporated. Head movement capabilities of individual users can vary significantly based on the severity of their injuries. A Gain parameter α enables customization of movement sensitivity (slope of transfer function) based on an individual's movement capabilities. The transfer function is described by Equation (25) and visualized in Figure 18.

$$\begin{bmatrix} Mouse_{x|t} \\ Mouse_{y|t} \end{bmatrix} = \begin{bmatrix} Mouse_{x|t-1} \\ Mouse_{y|t-1} \end{bmatrix} + \alpha \begin{bmatrix} Roll_t - \lambda_{roll} \\ Pitch_t - \lambda_{pitch} \end{bmatrix} \tag{25}$$

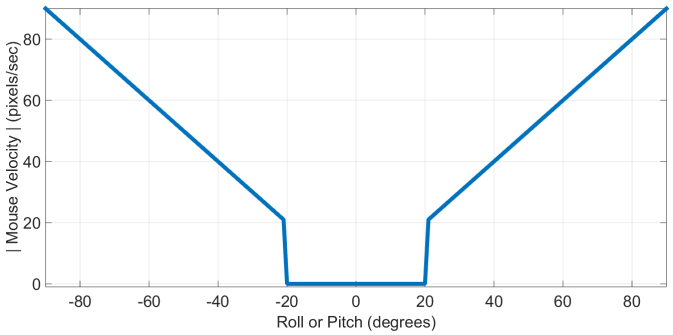

Figure 18: Transfer function translating angular displacement to mouse cursor velocity. Threshold $\lambda_{roll,pitch} = 20°$ and $\alpha = 1$. Roll and Pitch are symmetrical in this case and have the same transfer function.

3.3 Speech Recognition (SR)

The mTDS currently integrates the Dragon Naturally Speaking v13 (Nuance, Burlington, MA, USA) speech recognition engine owing to its low cost and multi-platform support. The mTDS software architecture however, retains the flexibility to use other commercially available speech recognition engines.

3.4 Discussion

Emulated tongue data using the four-dimensional robot allows us to find the effect of changing the magnet position, orientation, and combination of both to the performances of the different algorithms. It can be observed that KNN-9 which was used in the previous versions of TDS [51], performed significantly low regarding accuracy because of two possible reasons. First, the amount of training data used in this experiment is 3.3× less (total 700 s before vs. 210 s) than the previous work [51]. Second, the bad data point is not removed manually using 3D principal component analysis (PCA) unlike the previous work. The manual training data removal often resulted in erroneous training model and indeed a tedious process. In this work, no data is removed manually during the training or testing process.

SVM (RBF) under-performed due to the over-fitting of the training data. Test data-sets were completely different from the training data-sets that might result in such an outcome. However, due to orientation variation, the performance of the SVM (RBF) improved compared to position variation. LR also resulted in very different outcome due to magnet position vs. orientation variation. In a real scenario, a participant might vary the magnet orientation or position slightly during training vs. testing. Thus an algorithm such as LR is not suitable because of inconsistent performance due to tongue position vs. orientation variation.

It can be noticed that the SVM (RBF) performance reduced due to the change of both orientation and position of the magnetic tracer whereas LR performance improved compared to only orientation variation. SVM(RBF) overfitted the training data, but LR added more orientation variation in the training model which might result in the better command prediction accuracy for the test data. KNN performance also degraded in case of both position

and orientation variation. SVM (6D) under-performed compared to SVM(Linear) by 3.6% regarding accuracy, 8.5% in sensitivity, and 0.9% in specificity. This result implies that using an additional reference magnetometer to attenuate EMF interferences in SVM(Linear) indeed resulted in better performance compared to SVM(6D). It is also expected to obtain better performance using SVM(Linear) while speaking as it labels speech as Nc/ resting tongue command.

The performance of the classification using a KNN-9 algorithm for human data is also low compared to other options because of the same reason discussed above. KNN and SVM(Linear) have shown similar performances, but SVM (Linear) has a slightly better sensitivity (0.8%). SVM (RBF) did not perform well as opposed to SVM (Linear) because of overfitting the training data. It also requires more computation as opposed to SVM (Linear). SVM (6D) performs similar but has less accuracy (1%) as opposed to SVM (Linear). SVM (6D) requires less computation and one less magnetometer. However, it has a MidasTouch problem due to using differential EMF attenuation method. Although LR is computationally less expensive [55], it shows reduced performance regarding both accuracy and sensitivity compared to SVM (Linear).

Finally, a comparison is made regarding assigning tongue commands vs. resting command. SVM with linear kernel (SVM(Linear), SVM(6D)) shows better performances regarding assigning false positive commands which was expected. Using a reference magnetometer improved the performance of the classification algorithm by a little. However, the difference is more evident in the emulated tongue data while changing the orientation of the magnet. Changing the orientation of the magnet emulates the magnetic field generated by a user during speaking.

It is expected to find inconsistency between the performances of different algorithms using robot emulated vs. actual human tongue data, because the robot data for training and testing were never the same. Thus our goal was to find the best-performed algorithm despite any variation (magnet position, orientation, or both). Human data were also affected by the experience of a particular participant, the shape of the face and so forth. However, the trend of the result is similar for both emulated vs. human tongue data. it indeed

provided us more insight to find out which algorithm is more vulnerable to magnet position vs. orientation variations.

The proposed algorithm is only compared with the previously implemented options, with and without a reference magnetometer, and different kernels of SVM. A more computationally efficient, more accurate algorithm which is SVM (Linear), finally implemented in mTDS Headset. The SVM (6D) is more suitable for the intraoral tongue drive system (iTDS) [31, 59] where reference magnetometer is not feasible to use in the small oral cavity. SVM(Linear) will also allow mTDS headset to interface with assistive robots directly for rehabilitation purposes [60].

3.5 Conclusion

In this work, six different algorithms are compared to find the best-performed option for a limited resource wearable embedded hardware (mTDS). An optimized mathematical implementation of the algorithm is also proposed which is validated by both emulated and human participants tongue movement data. This algorithm will allow the mTDS to process tongue and head commands in-system (Headset) and interface with various devices without a need of any processing unit to control devices like smartphones, computers, wheelchairs, smart-homes and so forth.

CHAPTER 4

EVALUATION (ABLE-BODIED PARTICIPANTS)

The evaluation studies involved with the able-bodied participants first to compare the the performances with the gold standards. The able-bodied participants' studies are presented in the following sections.

4.1 Computer Study

A multitask study was then designed to help with more in-depth understanding of multimodal input systems for human computer interaction. We noted that a combination of different modes, either sequentially or simultaneously, has a significant effect on the user performance, and is likely to affect both accuracy and speed.

In this study, the results of a comparative and quantitative study are reported between a multimodal (mTDS) and unimodal (TDS) assistive technology, when used hands-free by able-bodied participants against KnM as the gold standard. Table 7 summarizes four tasks in this study and their corresponding performance measures. More specifically, the study was designed to address questions, such as:

- What is the throughput of the mTDS while proportional head movement and discrete tongue commands are combined in a center-out tapping task? Is a multimodal AT faster or more accurate than its unimodal counterpart? How do these systems perform compared to the KnM combination?

- How does proportional head-controlled HID perform in terms of speed and accuracy in a high constraint maze navigation task compared to a discrete tongue-controlled cursor movement, limited to cardinal movements?

- How does a head-controlled HID perform in a less constrained task, such as playing a game? Does the multidirectional proportional cursor movement help users score more than a switch-based controller?

- How does mTDS perform in a complex daily task, such as preparing and sending an email when multiple input modalities are combined vs. a single input modality? What is the typing speed and accuracy compared to KnM among the same group of participants?

4.1.1 Setup

The user interface (UI) for this study was developed in LabVIEW environment to provide participants with graphical input and collect usage data. All experiments were conducted on a desktop PC (Intel i7-5500U, 2.4 GHz, 16 GB memory) with two screens, monitor-1 (22" 1920 $\times$ 1080 LCD) for the participant to do the tasks and monitor-2 (15.6" 1920 $\times$ 1080 LCD) for the operator to switch between the experiments and observe their progression, as shown in Figure 19. The purpose of each task, given instructions, input devices, and performance measures are summarized in Table 7.

4.1.2 Protocol

The study was approved by the Institutional Review Board (IRB) at Georgia Institute of Technology. Fifteen able-bodied student volunteers (8 males and 7 females), ages 23-33 years old, 1 experienced and 14 naive with respect to TDS and mTDS, 1 native English speaker and 14 non-natives participated in this study. The study involved one session per subject, which was divided into three parts (preparation, training, and execution) and limited to 3 hours. To be prepared consistently, participants watched an instructional video, introducing them to TDS, mTDS, various input modalities, and all the tasks. The preparation was continued with a small disk-shaped magnetic tracer ($\phi 4.8mm \times 1.5mm$) from K&J Magnetics (Jamison, PA) being attached to the subject's tongue, $\sim 1cm$ from the tip, using cyanoacrylic adhesive, after drying the tongue with a paper towel, similar to [43]. The magnetic tracer attachment in this study was temporary, but at the same location where the tongue piercing should be done for long term use of the mTDS.

In the training part, the mTDS headset, which was also used as TDS input by disabling its inertial sensor and microphone, was calibrated to attenuate the earth's magnetic field (EMF). The speech recognition engine, Dragon Naturally Speaking (Nuance, Burlington,

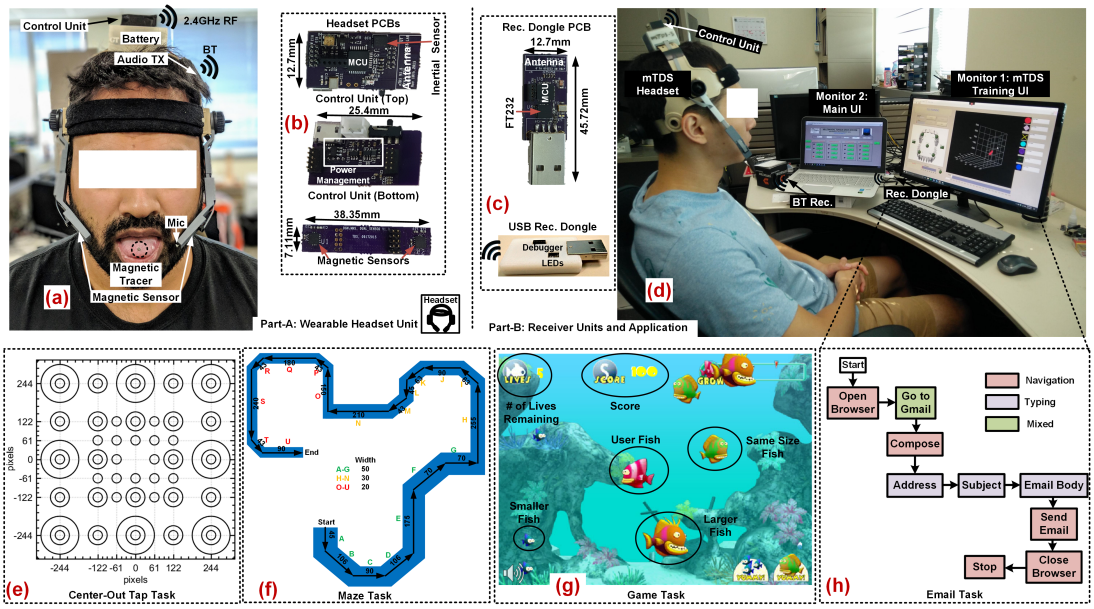

Figure 19: Part-A: mTDS headset (a) includes 4 magnetic sensors (2 on each pole) to measure the magnetic field generated by magnetic tracer attached to user's tongue. Accelerometer and gyroscope on the control unit for head tracking and a microphone with Bluetooth transmitter to send audio signal to PC. PCBs of the control and sensor units are shown in (b). A USB dongle and PCB are shown in (c), receives the sensor data from control unit. Part-B: Experimental setup (d) with the subject sitting $\sim 1m$ away from the 22" LCD monitor. Center-out tapping task targets (e), maze navigation task (f), indicating the lengths and widths of the trace in pixels, and color-code representing the difficulty of that segment, Fish Tales game task (g) requiring 2D navigation, and email sending task (h) in the form of color-coded steps. Subtasks c in the same window are shown in the same row of the block diagram.

MA) [61], was also calibrated, and tongue commands were trained following a procedure similar to [62]. In this study, tasks (center-out tapping, maze, game, or email) and input devices (TDS, mTDS, or KnM) were randomly selected for each subject to prevent any potential bias.

The following three input methods were used to compare between the devices:

4.1.3 Keyboard and Mouse (KnM)

A standard QWERTY Keyboard and a 3-button mouse with a roller were used as the gold standard. All participants were proficient in using these HIDs.

4.1.4 Tongue Drive System (TDS)

TDS uses four 3-axial magnetic sensors (LSM303D, STMicroelectronics, Switzerland) mounted symmetrically on two arms of a headset, to be held near the cheeks, as shown in Figure 19. They measure the changes in the magnetic field generated by the magnetic tracer on participant's tongue as it is placed in a set of seven user-defined locations in the mouth, each of which represents a discrete command [63]. A control unit, MCU (Figure 19), consisting of a microcontroller (CC2510, Texas Instruments, Dallas, TX) with built-in 2.4 GHz transceiver packetizes raw magnetic sensor data. Sensor data is collected by the MCU using a serial peripheral interface (SPI). The MCU transmits data packet to a PC via a wireless USB dongle (Figure 19). Sensor raw data is then processed to attenuate the EMF and classified using support vector machine (SVM) to generate the tongue commands, as discussed in [64]. The seven tongue commands are used to move the mouse pointer (Up, Down, Left, Right), issue button clicks (Left-Select, Right- Select), and the Resting (or no-operation) command, which corresponds to the tongue resting position and anywhere on the sagittal plane. Text entry with TDS was done via a commercial onscreen keyboard, known as Click-N-Type [65].

4.1.5 Multimodal Tongue Drive System (mTDS)

The mTDS adds head tracking and speech recognition capabilities to the TDS headset using a 9D inertial sensor (LSM9DS1, STMicroelectronics) and a microphone with its Bluetooth transceiver in the control unit, respectively. Head tracking is used to move the cursor in any direction at the desired speed based on the user's head motion (pitch and roll), as explained in [63]. Cursor movement speed is proportional to the user's head tilt. The mTDS also uses three tongue commands: Left-Select, Right-Select, and Resting, out of the seven TDS commands are used to incorporate mouse button clicks, while avoiding undesired mouse clicks when the tracer moves in the sagittal plane, for instance, while speaking. The mTDS is unique by supporting simultaneous use of two out of three modalities, which are among the key remaining abilities of most individuals with tetraplegia [62].

Four tasks were selected to be done using three different input devices which are described in the following sections:

Table 7: Tasks, input methods, and performance measures

Tasks	Center-out Tapping	Maze	Game	Email
Purpose	Pointing device evaluation	Constraint navigation evaluation	Unconstrained navigation evaluation	Complex task evaluation
Instructions	Reach target as fast as possible and select accurately	Navigate as fast as possible by keeping cursor inside the track	Score as much as possible within the time limit	Complete the task fast with the minimum typing errors
Time limit	N/A	2 min (each round)	2 min (each round)	5 min (each round)
TDS	Tongue (navigation and click)	Tongue (navigation)	Tongue (navigation)	Tongue (navigation, click, type)
mTDS	Tongue (click), head movement (navigation)	Head movement (navigation)	Head movement (navigation)	Tongue (click), head movement (navigation), speech (type)
KnM	Mouse (navigation and click)	Mouse (navigation)	Mouse (navigation)	Mouse (navigation and click) and keyboard (type)
Performance Measures	Throughput, Error rate, Missed targets, Path efficiency, reaction time	Navigation throughput, Completion time	Game Score	Completion time, % task completion, % typing accuracy

TDS: Tongue Drive System, mTDS: multimodal Tongue Drive System, KnM: keyboard and mouse combination.

4.1.6 Center-out Tapping Task

This standardized task is designed to evaluate the performance of pointing devices using Fitts's law [66]. During the task, round targets with different diameters (30, 60 and 122 pixels) and distances (61, 86, 122, 173, 244, 273, and 345 pixels) from the center of the screen randomly appeared, one at a time, as shown in Figure 19. Participants were instructed to move the cursor from the center towards the target and click as close as possible to the center of the target. A new random target appeared upon clicking the previous one. Each round included 48 random targets, which were located either in cardinal or ordinal direction from the center.

4.1.7 Maze Navigation

The on-screen maze navigation task in this study evaluates cursor movements within a highly constrained and predefined path using head movements vs. tongue commands. Users moved the cursor from a designated starting point to an endpoint through a maze, while trying to stay within the track boundaries, as shown in Figure 19. The width of the track (in pixels) changed the task difficulty level, which was increased by reducing the width from 50 to 30, and eventually to 20 pixels, as the user approached the endpoint. Each maze was divided into 21 segments. The transition from one segment to another required a change in direction

by $45°, 90°, 135°, 225°, 315°$ relative to the previous segment. Task duration was limited to 2 minutes for each round. Maze layout was selected randomly from 8 patterns with similar difficulty levels (length: 2262 ± 48 pixels) to prevent adaptation bias.

4.1.8 Playing a Game on PC

This task evaluates cursor navigation using head movements vs. tongue commands in conducting a less constrained and somewhat random but goal-oriented task, such as playing a computer game. A popular PC-based Flash game, called "Fish Tales" (Figure 19) was used to evaluate the performance of different input methods because of its simple rules and effective use of all directional commands. This 2-minute task evaluated the cursor navigation capability and agility in response to random game events, which unlike maze navigation were not predefined. The goal in this game is for the player's fish (red) to catch and eat smaller (blue) and same size (green) fish, which randomly entered the screen and also moved in random directions, to grow bigger, while keeping away from larger fish (brown). Players earned points upon catching fish and lost a life if they were caught by the larger fish. Upon reaching to the next level both speed and number of larger fish increase, thus making the game more difficult even though the player's fish also grows. Participants were promoted to harder levels until they lost all five lives or ran out of time.

4.1.9 Sending an Email

This task evaluates the HIDs in a complex and fundamental computer access task that utilizes cursor navigation, clicking, and text entry, in a combined or sequential manner. Participants were asked to compose an email with a randomly selected content, chosen from 60 sample texts. The length of each email was 2-3 sentences with comparable length and complexity (24.8 ± 4.7 words). The task was constrained to 5 minutes, and the same browser (Google Chrome) and email client (Gmail) were used with fixed sender and receiver addresses to minimize inconsistencies among subjects. The email client was signed in for everyone to bypass entering a password, consistently for all users. Figure 19 shows the optimal workflow, which was not enforced, with the color-coded modalities needed to complete different parts of the task.

In the following sections, a set of widely-accepted performance measures are defined and used to quantify the measurement results within the context of this study.

4.1.10 Center-out Tapping Task

4.1.10.1 Throughput (TP) (bits/s)

Quantifies information transfer rates from a pointing device to PC. It is the ratio between the index of difficulty (ID) to the average movement time (MT) with the same condition [66,67]. The final mean of mean throughput is calculated from [67],

$$TP_{avg} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{m}\sum_{j=1}^{m}\frac{ID_{ij}}{MT_{ij}}) \tag{26}$$

where m is the number of targets shown for a participant and n is a total number of participants. ID is defined by Shannon's formula using [68],

$$ID = log_2(\frac{D}{W} + 1) \tag{27}$$

where D is the distance from the starting point at the center of the screen to the center of the target and W is the diameter of the circular target, shown in Figure 19. ID is higher for smaller and farther targets that are more difficult to reach than larger and closer targets.

4.1.10.2 Reaction Time (RT)

Represents the time between when a target was shown and when the participant started to move the cursor towards it [69]. Final reaction time is reported from the mean of mean value,

$$RT_{avg} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{m}\sum_{j=1}^{m}RT_{ij}) \tag{28}$$

4.1.10.3 Error Rate (ER)

ER is the percentage of error in selecting a target. ER% is found from the ratio of sum of error selections to the total number of selections [37],

$$ER_{avg} = \frac{\sum_{j=1}^{x} Error\,Selection_j}{\sum_{i=1}^{y} Selection_i} \times 100, \tag{29}$$

where x is the total number of selections outside of the target and y is the total number of selections for all subjects in 3 rounds. Here, "Error selection" parameter is either '1' or '0' based upon the wrong or correct selection, respectively.

4.1.10.4 Missed Target (MsT%)

MsT(%) is defined by percentage of targets, for which the movement is less than half the nominal distance, D, of those targets, or which selection was more than twice the target width, W, away from the center of the target [70–72]. These targets are neither reported in ER nor in TP calculation. MsT% is calculated from,

$$MsT_{avg} = \frac{\sum_{j=1}^{q} missed\,target_j}{q} \times 100, \tag{30}$$

where q is a total number of targets. A missed target is either '1' or '0' based on being missed or not.

4.1.10.5 Path Efficiency (PE%)

is the ratio of the euclidean distance from the start point to the target center to the actual distance navigated by the cursor in percentage [37],

$$PE_{avg} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{m}\sum_{j=1}^{m}(\frac{eucl.\,distance}{nav.\,distance} \times 100)_{ij}) \tag{31}$$

4.1.11 Maze Navigation Task

4.1.11.1 Maze Throughput (TPm) (bits/s)

Maze throughput combines both speed and accuracy of the input device during cursor navigation in a constrained path and can be a precursor for wheelchair navigation. Mean of mean maze throughput [73] was reported to quantify subjects' performance using different HIDs, which is defined as,

$$TPm_{avg} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{t}\sum_{j=1}^{t}(\frac{ID_P P_{in}(1-P_{out})}{CT})_{ij} \tag{32}$$

where P_{in} and P_out are percentages of the distance that the mouse pointer has traversed inside and outside of the maze, respectively [73]. Maze completion time, CT, measures the time required for a participant to navigate from start to endpoint, and t is the total number of trials. Path index of difficulty, ID_P in bits, is calculated using Shannon's formula from the length (L) to width (W) ratio of the maze segments,

$$ID_p = \sum_{i=1}^{s} log_2(\frac{L_i}{W_i} + 1), \tag{33}$$

where s is the number of maze segments.

4.1.12 PC Game Task

4.1.12.1 Game Score (GS)

Captures both speed and accuracy of the subjects across HIDs when playing the game. Mean of the mean score is reported as,

$$Score_{avg} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{t}\sum_{j=1}^{t} score_{ij} \tag{34}$$

4.1.13 Sending Email Task

4.1.13.1 Task Completion Time (TCT) (min)

TCT Is the time required to complete the email task, and only captures the subject's speed according to [63],

$$TCT_{avg} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{t}\sum_{j=1}^{t} TCT_{ij} \tag{35}$$

4.1.13.2 Percentage Task Completion (PTC)

Represents percentage of the task that was completed by a participant based on a scoring system [63]. Email task was divided into the following subtasks and their corresponding points (0 5): unable to open the browser (0), opened the browser (1), typed in the From

email address (2), typed subject of the email (3), partially typed email body (4), completed sending the email (5). These points were then converted to a percentage using,

$$PTC_{avg} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{t}\sum_{j=1}^{t}(\frac{point\,achieved}{5} \times 100)_{ij} \tag{36}$$

4.1.13.3 Percentage Typing Accuracy (PTA)

PTA finds how accurately participants typed an email [63]. PTA is the ratio of the number of correct words typed to the total number of words shown,

$$PTA_{avg} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{t}\sum_{j=1}^{t}(\frac{\sum \#\,of\,correct\,word\,typed}{\sum \#\,of\,word\,shown} \times 100)_{ij} \tag{37}$$

A total number of correct words only include the words shown and typed as a reference. Misspelled words were not included in the PTA calculation.

4.1.14 Results

Four tasks were conducted using three input devices (a total of 12). The first out of four rounds of each task was considered as a practice, and the following three were analyzed for performance evaluation. Mean of mean performances are summarized in Table 7. One-tail t-tests ($\alpha = 0.05$) were conducted to find the difference between the mean performances of mTDS and TDS. The Null hypothesis, H_0 of the test is: there is no difference between the average performance of mTDS and TDS. The statistical analysis results; t-value, t_{cr}-value, degree of freedom (df), p-value, and outcome of the t-test, are also summarized in Table 8.

4.1.14.1 Center-out Tapping Task

The mTDS average throughput among all subjects was 0.84 bps, which was 20.1% of the mouse, where head movements and tongue commands were used sequentially to move the cursor and select a target. TDS throughput, on the other hand, was 0.94 bps, which was 22.4% of the mouse. The mouse throughput found in this study, 4.2 bps, is within the previously reported results, which varies from 3.5 to 4.5 bps [67], [74], validating the experimental procedure for this task. Movement time (MT) vs. index of difficulty (ID) for all

Table 8: Task performance (mean ± sem) and statistical analysis results

			KnM	mTDS	TDS
Center-out Tapping Task					
TP (bits/s)		Cardinal	3.97 ± 0.04	1.07 ± 0.02	1.21 ± 0.04
		Ordinal	4.32 ± 0.03	0.70 ± 0.01	0.76 ± 0.01
		Overall	4.19 ± 0.03	0.84 ± 0.01	0.94 ± 0.02
RT(s)		Cardinal	0.20 ± 0.00	0.63 ± 0.02	0.90 ± 0.04
		Ordinal	0.18 ± 0.00	0.53 ± 0.02	0.54 ± 0.03
		Overall	0.19 ± 0.003	0.57 ± 0.01	0.69 ± 0.02
ER(%)		Cardinal	2.27	8.45	9.72
		Ordinal	3.85	17.24	20.05
		Overall	6.12	25.68	29.77
MsT(%)		Cardinal	0	0	0
		Ordinal	0.09	6.81	32.36
		Overall	0.09	6.81	32.36
PE(%)		Cardinal	57.54 ± 0.90	62.49 ± 1.15	65.43 ± 1.19
		Ordinal	60.27 ± 0.61	48.59 ± 0.65	49.99 ± 0.67
		Overall	59.27 ± 0.51	53.87 ± 0.61	56.33 ± 0.66
Maze Task					
TPm (bits/s)			2.11 ± 0.11	0.35 ± 0.03	0.46 ± 0.05
Game Task					
GS			1302 ± 22	498 ± 25	289 ± 22
Email Task					
TCT(min)			1.22 ± 0.05	2.41 ± 0.14	5.00 ± 0
PTC(%)			100 ± 0	98.67 ± 1.33	65.78 ± 3.22
PTA(%)			98.06 ± 0.50	78.15 ± 2.77	17.74 ± 2.37
Statistical Analysis (one-tailed t-test)					
		t	t_{cr}	df	p	H_0
Center-out Tapping Task						
TP (bits/s)	Cardinal*	2.13	1.76	14	0.02	rejected
	Ordinal	0.94	1.76	14	0.18	accepted
	Overall*	2.08	1.76	14	0.03	rejected
RT (s)	Cardinal*	4.04	1.76	14	6e-4	rejected
	Ordinal	0.17	1.76	14	0.43	accepted
	Overall*	2.35	1.76	14	0.02	rejected
ER (%)	Cardinal	1.38	1.76	14	0.09	accepted
	Ordinal	1.51	1.76	14	0.08	accepted
	Overall	1.45	1.76	14	0.08	accepted
MsT (%)	Cardinal*	4.49	1.76	14	2e-4	rejected
	Ordinal*	6.48	1.76	14	7e-6	rejected
	Overall*	6.49	1.76	14	7e-6	rejected
PE (%)	Cardinal	0.56	1.76	14	0.29	accepted
	Ordinal	0.65	1.76	14	0.26	accepted
	Overall	0.79	1.76	14	0.22	accepted
Maze Task						
TPm (bits/s)		1.35	1.76	14	0.10	accepted
Game Task						
*GS		10.20	1.76	14	3e-8	rejected
Email Task						
*PTC(%)		8.15	1.76	14	1e-6	rejected
*PTA(%)		12.9	1.80	11	2e-8	rejected

TP: Throughput, RT: Reaction Time, ER: Error Rate, MsT: Missed Target, PE: Path Efficiency, TPm: Maze Throughput, GS: Game Score, TCT: Task Completion Time, PTC: Percentage Task Completion, PTA: Percentage Typing Accuracy, t: one sided t-value, tcr: critical t-value with $\alpha = 0.05$, df: degree of freedom, P: p value, H0: Null hypothesis.*statistically significant

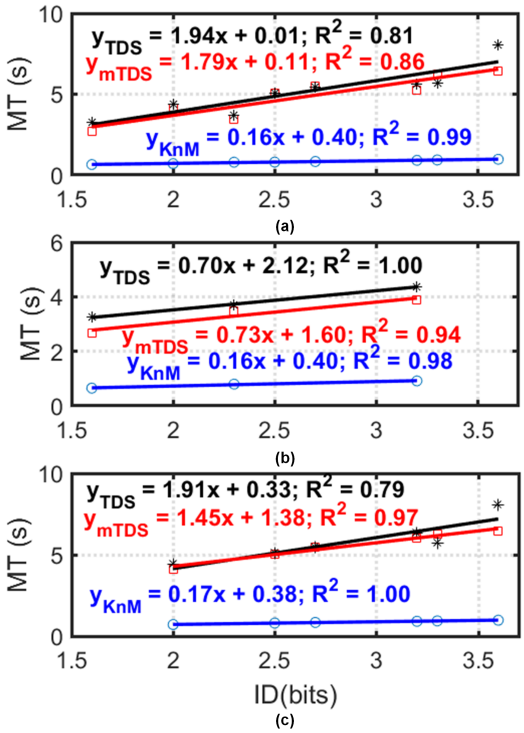

Figure 20: Goodness of fit curve for MT vs. ID (a) all targets, (b) cardinal targets, and (c) ordinal targets.

input devices are shown in Figure 20. The goodness of fit, R2, using TDS is 0.81, which is lower than both mTDS (R2 = 0.86) and mouse (R2 = 0.99). The reason for a low R2 for TDS might be because the cursor can only be moved in cardinal directions with this device. Results from the cardinal and ordinal targets are analyzed and plotted separately in Figure 20, respectively, to test this assumption further. The goodness of fit, R2 for MT vs. ID linear model for cardinal and ordinal targets using TDS are 1 and 0.79, respectively. Unlike TDS, R2 is closer to 1 for both cardinal and ordinal targets (Figure 20) using mTDS and mouse, because of their proportional and multidirectional navigation capability.

Throughputs for cardinal and ordinal targets using mTDS were 1.07 bps and 0.7 bps, which were 26.9% and 16.2% of the mouse, respectively. Using TDS, they were 1.21 bps and 0.76 bps, which were 30.5% and 17.6% of the mouse, respectively. The difference between average throughput of mTDS and TDS is statistically significant, as shown in Table 8.

The average reaction times (RT) were 0.57 s and 0.69 s, which were 3 and 3.6 times that of a mouse, using mTDS and TDS, respectively. They were 3.1 and 2.9 times that

of a mouse, for the cardinal and ordinal targets, respectively, using mTDS, and 4.5 and 3 times that of a mouse using TDS. Average RT for the cardinal targets was more than the ordinal target using all input devices, which is counter-intuitive. This result indicates that participants took more time to initiate a command to move the cursor to reach cardinal vs. ordinal targets. The average RT of mTDS was less than TDS, which is also statistically significant, as shown in Table 8.

Overall error rates (ER) were 25.68% and 29.77%, which were 4.4 and 4.9 times that of a mouse, using mTDS and TDS, respectively. They were 3.7 and 4.5 times that of a mouse, for cardinal and ordinal targets, respectively, using mTDS, and 4.3 and 5.2 times that of mouse using TDS, respectively. These results show that the ER of both mTDS and TDS were higher for ordinal targets. However, it was less using mTDS than TDS. The average ER difference between mTDS and TDS is not statistically significant though (Table 8). It is to be noted that the same tongue command (Left Select) was used to click on a target for both devices.

Using mTDS 6.8% of the targets were missed, all of which were located in the ordinal direction from the center. However, 32.4% of the targets were missed using TDS, which is 25.6% more than mTDS. All missed targets with TDS were also in the ordinal direction. The average missed target difference between mTDS and TDS is statistically significant (Table 8). Hence adding more modalities in mTDS, helped participants reach more targets compared with TDS. The average missed target for mouse was only 0.1%.

Overall path efficiencies (PE) were 53.87% and 56.33%, which were 5.4% and 2.9% less than that of a mouse, using mTDS and TDS, respectively. The PE of mTDS was 4.9% more and 11.7% less than that of a mouse for cardinal and ordinal targets, respectively. They were 7.9% more and 10.3% less than that of a mouse for TDS. PEs for cardinal targets were more than that of a mouse using both the mTDS and TDS. The average PE difference between mTDS and TDS are not statistically significant (Table 8).

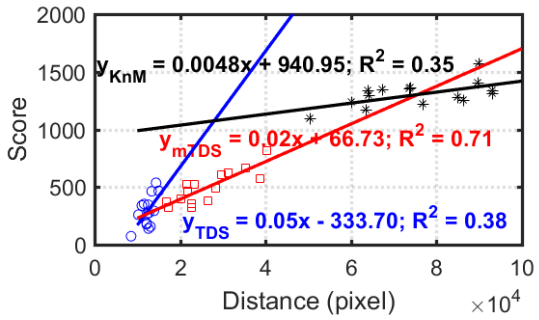

Figure 21: Distance navigated vs. game score model for different input methods.

4.1.14.2 Maze Navigation

Overall maze navigation throughputs were 0.35 bps and 0.45 bps, which were 16.6% and 21.8% of a mouse, for mTDS and TDS, respectively. TDS throughput was 31.4% higher than mTDS. However, in this task only head movement of mTDS was used to control the cursor. The difference of the mean maze navigation throughput between mTDS and TDS is not statistically significant (Table 8). This result implies that for constrained cursor navigation task, discrete tongue commands are more suitable than proportional head movement in terms of both speed and accuracy.

4.1.14.3 PC Game

Unlike maze navigation, in the game task participants were able to move the cursor more freely, while still following the game objective. Using proportional head movement to control the cursor, the average score using mTDS was 498, which was 72.3% higher than TDS. mTDS and TDS scores were 38.3% and 22.2% that of a mouse, respectively. Distance vs. score graphs of different input devices are plotted in Figure 21 to find out the correlation between the traversed cursor distance and score. Linear distance vs. score models are found for each input device. The correlation between score and distance is higher using mTDS (0.71) compared to TDS (0.38) and mouse (0.35). These graphs show that the participants who moved the cursor more scored higher. Statistically, the mean mTDS score is higher than TDS (Table 8).

4.1.14.4 Email Task

The email task combines all modalities of the mTDS (head movement, tongue command, and speech recognition), which are simultaneously available to the user. Average mTDS email task completion time (TCT) was 2.41 min, which was 1.98 times that of KnM. The typing accuracy was 78.15%, which was 19.9% less than that of KnM. None of the participants were able to complete the task within the maximum 5-minute limit using TDS. The average percentage task completion (PTC) using TDS was 65.8%, which was 32.9% less than mTDS with 98.1% completed tasks. This result indicates that combining all modalities considerably improved participants' ability in completing complex tasks, such as preparing and sending an email. The typing accuracy using mTDS was 60.4% higher than that of the TDS. Despite participants' accent, typing via speech recognition was significantly more accurate than the on-screen keyboard using tongue commands. Statistically, mean PTC and PTA, using mTDS are significantly higher than that of TDS, as shown in Table 8.

4.1.15 Discussion

This study evaluates three input devices to access a computer while doing tasks with different levels of difficulty, all in a single session. The study also compares single vs. multimodal (email and center-out tapping tasks) and discrete vs. proportional control (all tasks) nature of computer interactions. During the center-out tapping task using mTDS, proportional head movement was used to navigate the cursor and discrete tongue command to click. mTDS takes advantage of the proportional, multidirectional nature of the head motion and discrete nature of the tongue commands as substitutes for hand/finger movements and button clicks in a mouse or touchpad, respectively. Classical Fitts's law measures were used to find the throughput of the input devices that generated better MT vs. ID linear correlation for this study compared to MT vs. IDe (modified index of difficulty) [37]. The throughput was higher using tongue commands than head-based navigation for this task. However, combining both modalities enhanced participants' ability to reach more targets. Participants have less missed targets while using both the head and tongue combination than when using the tongue alone. Users move their tongues from directional commands to

the neutral position and then issue the click commands to select the target using the tongue. In this process, participants issued more false positive commands, which resulted in more missed targets. These results also show that participants were able to coordinate navigation and click tasks better while using head-tongue combination over the tongue alone. The error rate was higher for both mTDS and TDS than mouse because the same tongue command (LS) was used to select a target. We anticipate this difference to be less over time with participants' learning to use the tongue commands more rapidly and accurately.

Maze throughput of TDS is higher than mTDS, where tongue as opposed to head was used to control the cursor. Despite better performances, tongue commands are discrete and can be used to move the cursor only in cardinal directions. This result also implies that discrete tongue commands might be safer for driving a wheelchair than head movements, particularly when the end user is still learning how to use the mTDS. Switching to discrete tongue commands to drive a wheelchair would also reduce the possibility of issuing undesired commands or losing control due to involuntary inertial head motion while driving in a rough terrain.

The game score using mTDS being higher than the TDS reveals a real benefit of availability of proportional cursor control. Even though this is also a cursor navigation task, like the maze, the outcome is quite different because of the nature of the task requirements. A less constrained cursor movement was required for playing the game, compared to the maze task, which still meets the game objectives. It is also worth noting that using mTDS, the average distance traversed by participants while playing the game was 2.1 times that of the TDS. By combining all three modes of mTDS during the complex email task, the participants' performances were enhanced even further to the extent that the gap between the KnM and mTDS was significantly reduced, compared to other tasks. Whereas using a single modality (TDS), none of the users were able to complete the email task within the 5-minute time constraint.

Speech Recognition is used as a method of typing during the email task, while both head movement and tongue commands can also be used for this purpose via an onscreen keyboard. This is useful in scenarios, such as typing passwords, names, or emails, in which

Table 9: Fatigue and usability evaluation

Fatigue Questions		
Questions	**Yes**	**No**
Was your tongue tired?	6	9
Was your jaw tired?	3	12
Was your neck tired?	6	9
Were your shoulders tired?	5	10
Usability Questions (1: very difficult to 5: very easy)		
Questions	Score (mean ± std)	
How difficult is it to use head for cursor control?	3.7 ± 0.7	
How difficult is it to switch modes of mTDS?	4.3 ± 0.7	
How difficult is it to use mTDS over TDS?	4.6 ± 0.6	
How difficult is it to use mTDS over KnM?	2.6 ± 0.6	

mTDS: multimodal tongue drive system, KnM: Keyboard and mouse combination.

the user privacy is paramount, as well as in noisy environments, where speech recognition accuracy is low. Error correction was not applied to any of the tasks in this study.

Participants were asked questions regarding fatigue in parts of the body that are engaged in this study after the 3-hour session, and usability of each device, which results are shown in Table 9. The majority of participants did not feel fatigue. However, the impact of fatigue on the user performance needs to be considered. In the scale of 1, being very difficult, to 5, being very easy, the average difficulty to use head movements to control mouse cursor was 3.7. The average difficulty of switching between the modes of mTDS was 4.3. The difficulty in using the mTDS compared to TDS and KnM, were 4.6 and 2.6, on average, respectively.

There were several limitations in this study. All able-bodied subjects were naive with respect to TDS and mTDS, except for one subject, and their performances cannot be considered a real benchmark of the TDS or mTDS compared to the gold standard KnM combination, in which they were all proficient. This study was conducted only in one session, and no learning capability or measures were analyzed from this perspective. TDS is practically a subset of the mTDS, even though when using the mTDS, all possible tongue commands were not utilized. Two of the seven TDS commands were utilized to issue clicks, which may generate a bias towards mTDS in learning analyses of the acquired data.

It should be noted that while in this study we have demonstrated the availability and

simultaneous use of both proportional and discrete modalities, we have stopped short of specifically measuring the participants' proficiency in the simultaneous use of these modalities, which are necessary in doing complex computer interaction tasks, such as drag-and-drop, zoom in-and-out, multiple selections, hold and scroll, etc. This can be the subject of a future study.

Even though they did not use their hands, all subjects were able-bodied, as opposed to the target population for whom both TDS and mTDS have been designed. Nonetheless, this study reveals key aspects of combining different modalities, particularly discrete and proportional, in a modern assistive device, such as mTDS, while conducting computer access tasks at different levels of difficulty. A study with target population is underway, and future studies will include both sequential and parallel mode combinations of the mTDS to reveal the possibility of additional performance enhancements.

4.1.16 Conclusions

This study compared the performance of a multimodal vs. unimodal hands-free input system for human-computer interaction in four tasks with different levels of complexity. Using only discrete tongue commands to move the cursor generates 10.6% more throughput than head movement during the simple center-out tapping task. Proportional head-based cursor navigation combined with tongue commands, on the other hand, helped participants to reach 25.5% more targets with slightly fewer errors (4.1%). Tongue performed 23.9% better than head movement in navigating the cursor in a highly constrained maze. However, in a less constrained goal-oriented task, such as playing a PC game, proportional head-based navigation generated a considerably higher score (42.6%) than tongue-based navigation. Multiple combined modes enhanced the participants' abilities even further in complex tasks, such as generating and sending an email, by reducing the performance gap between the mTDS and KnM to be only two times slower and 20% less accurate, even though a large majority of the participants were naive in using to the former input device and proficient with respect to the latter.

Table 10: Tasks, input methods, and performance measures

Tasks	Game	Wheelchair Drive
Device under control	Computer	Wheelchair
Evaluation	Cursor navigation	wheelchair driving
Given Instructions	Score as much as possible within the time limit	Complete the task fast with the minimum of error
Time limit	2min (each round)	-
TDS	Tongue (Left, Right, Up, Down)	Unlatched, Latched, Semi-proportional
Joystick	-	4-directional control
Performance Measures	Game score	Completion time, Number of errors

TDS: Tongue Drive System, KnM: keyboard arrows.

4.2 Wheelchair Study

4.2.1 Protocol

Fifteen able-bodied volunteers (C01-C15), 8 males and 7 females between the ages of 18 to 35 years old, one of whom was experienced, one was intermediate, and 13 naive with respect to TDS, participated in this study. The study involved one session per subject which was divided into three parts (preparation, training, and execution) and limited to three hours. During the preparation, participants were introduced with the PC access and wheelchair driving tasks followed by signing a consent form. The preparation continued by attaching a small disk-shaped magnetic tracer ($4.8mm \times 1.5mm$, K&J Magnetics, Jamison, PA), to $\sim 1cm$ from the tip of the tongue using a cyanoacrylate tissue adhesive, similar to [43]. Attaching the magnetic tracer to the tongue tip is a temporary solution to test the efficacy of the system. However, for long-term usage, a participant will need to do a tongue piercing, and a magnetic tracer will be used as a stud to find the tongue commands.

During the training, the headset was calibrated (20 s process) to remove earth's magnetic field (EMF), and seven tongue commands were trained (10 s each command, total 70 s).

In the execution part, a PC task and three modes of wheelchair driving were done using tongue commands. PC tasks were compared with keyboard arrow keys and wheelchair driving with a joystick. Each task was repeated in four rounds, first as practice and rest three for evaluation. Table 10 shows the summary of the tasks and performance measures.

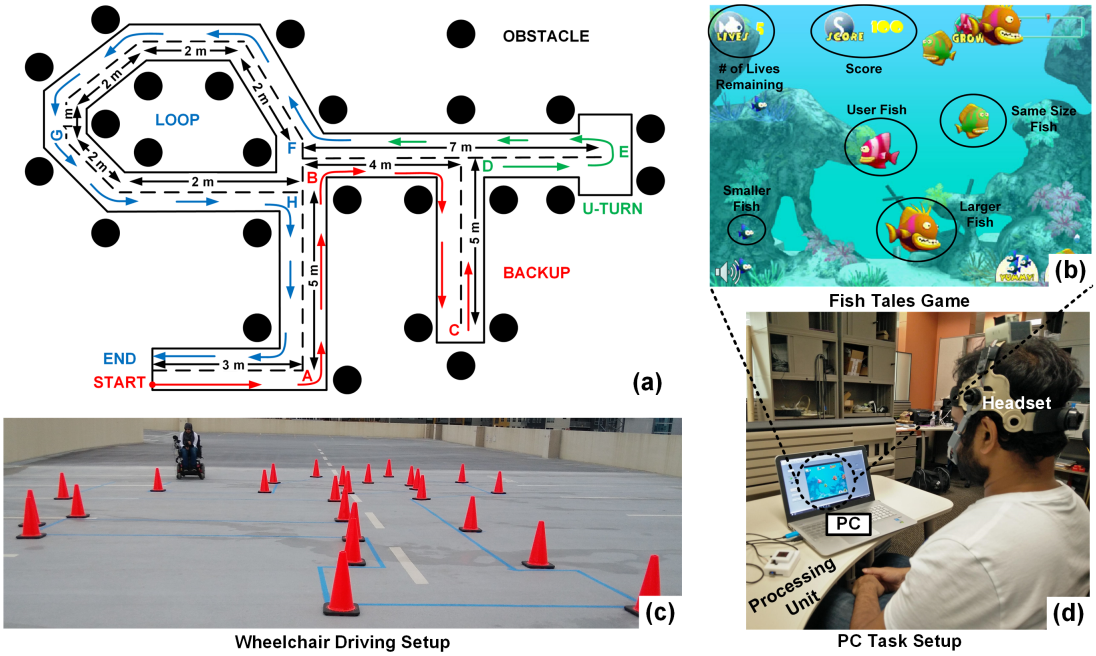

Figure 22: (a) A participant driving the wheelchair through the obstacle course, made of a 51-m track and 25 traffic cones, using the embedded TDS (b) Fish Tales PC game, where user navigates a fish (red) using (TDS/ arrow keys) to eat smaller (blue) or same size (green) fishes to score more and avoid larger fish (brown) to save life (c) Top view of the obstacle course (d) PC game setup.

4.2.2 Wheelchair Navigation Task

Participants drove the Q6000 wheelchair in a $\sim 51m$ long obstacle course, shown in Figure 22, which is identical to the one used in [23] for a valid benchmark, except for using 25 obstacles instead of 24 in the previous study. The track has a total of 13 left, and right turns, forward, backward, and reverse movements, as well as a U-turn, which are the scenarios that a TDS user might face while driving a wheelchair. The wheelchair navigation algorithm behaves differently depending on the selected mode of driving, out of a) unlatched, b) latched, and c) semi-proportional modes.

For unlatched mode, the values of two aforementioned vectors, linear and rotation, are gradually reduced or increased based on the issued command (UP, DOWN, LEFT, RIGHT), increasing or decreasing the speed of the wheelchair linear movement or rotation until they reach a predefined limit. The linear speed and rotation angle are implemented into 8 different equal levels, 0 to $1.61km/hr$ and 0 to $27°/s$, respectively, in the current study. If the user

returns the tongue to its resting position, both vectors return to zero bringing the wheelchair to a standstill. These vectors are also set to zero in the case of a lost connection between the TDS headset and the processing unit for safety.

For latched mode, the user needs to hold onto the UP command for 1 s to latch the wheelchair movement in a forward direction at minimum speed. Speed can then be increased by holding the tongue commands longer. In this mode, the wheelchair keeps going forward until the user issues a DOWN command. If the user holds the DOWN command for 1s from a stationary status, the wheelchair starts going backward. Backward speed vector amplitude also increases if the DOWN command is upheld until the backward speed limit of $1km/hr$ is reached. The user can turn the wheelchair left or right by issuing LEFT or RIGHT commands, respectively, either when the wheelchair is stationary or while it is moving forward. Rotation vector amplitude increases with the duration of the assigned LEFT/RIGHT tongue command. If the command changes to resting, the rotation vector is returned to zero.

For semi-proportional mode, only UP and DOWN discrete commands are detected by the machine learning algorithm. The amplitude of the rotation vector is determined proportionally to the amount of tongue deviation from the midline towards the left or right sides of the user mouth midline over the lips. The more the tongue moves to the left or right side of the mouth, the greater would be the absolute value of the rotation vector in that direction, which results in a sharper wheelchair rotation in that direction. The user can turn the wheelchair either when it is latched forward or when it is stationary. Going forward and backward would be the same as the latched mode.

Since most participants were a novice (never drove a wheelchair before), we set up the driving task from the easiest (joystick) to the most challenging (semi-proportional) mode. Therefore, instead of randomizing, all participants started the driving task using a joystick, in which case the speed, wheelchair motion, and rotation are proportional to the amount of controller deviation from the neutral position in the desired direction. One operator always walked next to the participant with the mechanical emergency stop button in hand to react quickly in the case of an emergency by shutting off the wheelchair. The same person tracked

Table 11: Wheelchair driving task results

Trial No.	**TCT (s) (mean ± std)**				**comp. JS/L**
	JS	**unL**	**L**	**SP.**	
0	135.5 ± 28	288.1 ± 41	203.8 ± 36	311 ± 100	1.5
3	114.1 ± 10	255.3 ± 35	190.7 ± 23	236.5 ± 78	1.67
Overall	119.1 ± 15	260.8 ± 35	196.2 ± 34	260.5 ± 74	1.65
I	15.79	11.38	6.42	24.15	
	# of Errors (mean ± std)				
0	1.7 ± 2.3	2.4 ± 2.2	3.8 ± 3.4	9.7 ± 7.6	2.23
3	0.5 ± 0.9	1.9 ± 1.6	2.8 ± 2.1	4.8 ± 4.4	5.6
Overall	1.2 ± 1.9	2.8 ± 2.6	4.6 ± 5.9	7.4 ± 6.9	3.8
I	70.58	20.83	26.32	50.51	
Statistical Analysis (paired one tail t-test, $\alpha = 0.05$)					
Learn.	$t(t_{cr})$	P	df	H_0	
JS (TCT)	-3.87(-1.76)	0.00085	14	**R***	
unL(TCT)	-3.51(-1.76)	0.0017	14	**R***	
L (TCT)	-1.24(-1.77)	0.1175	13	A	
SP. (TCT)	-1.77(-1.81)	0.053	10	A	
JS (Err.)	-1.77(-1.78)	0.05	12	A	
unL (Err.)	-0.64(-1.76)	0.26	14	A	
L (Err.)	-1.29(-1.78)	0.11	13	A	
SP. (Err.)	-2.08(-1.83)	0.034	9	**R***	

TCT: Task completion time, JS: joystick, L: Latched, unL: unlatched, SP.: Semi-proportional comp.: comparison, I: mean improvement (%), statistically significant*.

the timing and counted the number of driving errors, which included hitting an obstacle, moving one wheel out of the track, and moving two (×2 points) or more wheels out of the track. A second operator also recorded the timing and number of driving errors to minimize data collection errors.

4.2.3 Results

Overall average performances are reported from the mean of mean performances (game score, task completion time, and number of driving errors) of all participants. One-tailed Paired t-test ($\alpha = 0.05$) was done to find the ease of learning for the newly developed system from the first to last round of each task and driving modes. Comparison between arrow keys for the PC and TDS is carried out for the last trial of PC tasks (Game and Maze). One-way analysis of variance (ANOVA) was used to find the difference between TDS PWC driving modes regarding completion time and driving error. Finally, a post hoc test (Bonferroni) was

used to find the performance differences of different wheelchair driving modes using TDS. Outlier data was found and removed before statistical analysis using 2.2 times less and more than first to fourth quartile range of each round of the task performance metrics [75].

Table 11 shows the task completion time and number of errors, improvements over the trials, and statistical analysis results. The fastest overall average task completion time ($196.2s$) is achieved using TDS latched mode which is 60.7% of a joystick. Least overall average error (2.8) is achieved using TDS unlatched mode which is 2.3 times that of a joystick.

One-way analysis of variance (ANOVA) is done to compare between the TDS modes (unlatched, latched and semi-proportional). Null hypothesis, H0: mean task completion time (TCT) using three different TDS modes are equal, is rejected ($F = 7.13, df = 2, P = 0.0023$). Post hoc test (Bonferonni) shows that TDS Latched mode is significantly faster than unlatched mode ($p = 0.0021$) and mean difference is not significant between semi-proportional and latched or unlatched modes. For driving error, ANOVA did not reject the null hypothesis, H_0, of the mean number of driving errors using different TDS modes being equal ($F = 3.2, df = 2, P = 0.0518$). However, the p-value is very low. Therefore, the Bonferroni test was carried out to find whether the mean number of driving error differences between TDS modes was significant or not. Using unlatched mode, the mean number of driving errors was significantly less than semi-proportional mode ($p = 0.0474$). However, the mean number of driving errors is not significantly different between latched and unlatched or semi-proportional modes.

Improvements (joystick: 15.79%, unlatched: 11.38%, latched: 6.42%, and semi proportional: 24.15%) are found in average task completion times using different driving modes from first to the last trial. However, statistically, mean difference is only significant for the joystick and unlatched driving mode which is shown in Table 11.

Improvements (joystick: 70.58%, unlatched: 20.83%, latched: 26.32%, and semi proportional: 50.51%) are also found in average number of driving errors from the first to last trials. However, it is only statistically significant for the semi-proportional mode of driving.

4.2.4 Discussions

Since 86.67% (13 out of 15) participants had never driven the wheelchair before using either joystick or tongue commands, a statistically significant improvement is observed during first two driving methods, joystick and TDS unlatched mode.

Statistically, it can be concluded that Latched mode is fastest and unlatched mode is least erroneous among the TDS driving options. However, the fastest TDS PWC driving option (Latched) is 1.65 times slower and most accurate option (unlatched) is 2.3 times more erroneous than the gold standard, joystick driving. Unfortunately, this is not a wheelchair driving option for a person with a severe disability. It is important to make both of these modes available to TDS users because the unlatched mode is more suitable when driving in a confined space (less erroneous) even though the latched mode might be faster.

All participants were able to complete the semi-proportional mode during trial 0. However, 3 participants during trial 1 and 4 participants during trials 2 and 3 were unable to complete the task because of the 3-hour experiment time limit. We also observed that some participants had difficulties during wheelchair navigation using the semi-proportional mode. This mode is a combination of discrete tongue command classification for linear motion and a rather primitive proportional control algorithm based on magnetic field vector subtraction. To turn the wheelchair, subjects need to move the tongue to left or right over the lips, while avoiding the two discrete commands (UP/DOWN) that are defined very close to that space, i.e. by touching the tip of the tongue to the first premolar (left) and first premolar (right) teeth. If they issue UP/DOWN commands, the wheelchair stops instead of turning. Although their mean task completion time reduced from 311.80 ± 100.5 to 236.45 ± 78.7, the number of driving errors are higher (4.8 ± 4.4) compared to the other two modes (unlatched: 1.9 ± 1.6, latched: 2.8 ± 2.1) of driving during the trial 3. We concluded that novice participants had difficulty learning this method of driving. We have benchmarked the existing tongue-controlled ATs in Table IV, which shows that the embedded TDS can provide more control options and flexibility to individuals with tetraplegia compared to existing ATs. Wheelchair driving task performances (task completion time and number of errors) are also benchmarked using the same driving track. It shows 8.91% (previous: 207.7s error: 2.1 ± 2.5

to current: $190.7s$ error: 2.8 ± 2.1) faster average navigation time in the new embedded TDS performance over our previous study [23]. The performances are also compared with a study [41], conducted by the different research group, which adopted a similar track as described in [23]. This work shows better average velocity ($0.22 \sim 0.34m/s$) compared to ($0.18 \sim 0.29m/s$) in [41].

Limitation of this study compared to [23] was the convenient recruitment of able-bodied participants, as opposed to the potential TDS end users. However, the purpose of this study was to demonstrate the functionality of the embedded TDS and compare with the gold standards (arrow keys for PC, Joystick for wheelchair drive), which is only possible by recruiting able-bodied participants. This system has more features and robustness compared to other existing ATs. A follow-up multi-session study by recruiting end users is among our future research plans.

The novelty of embedded TDS is the inclusion of an independent low footprint processor to take care of the real-time sensor signal processing with an in-system visual feedback, various connectivity (2.4GHz RF, Bluetooth, WiFi) options, interfaces (PWC, PC) with a redesigned headset with electronics. It does not require to use multiple assistive devices to interact with different devices which require least assist from a caregiver or a family member. It also opens the door to interfacing with a wide range of devices like smartphones, IoT devices, TVs, garage doors, air conditioners, etc. all from a single platform, without interfering with the ongoing processes on the user's PC or smartphone, which further improves the TDS robustness, safety, and flexibility of usage. It is also possible to use the embedded TDS for rehabilitation purposes, instead of using a computer, to control a robotic arm [60]. The inclusion of the visual feedback allows the embedded TDS to operate without any PC/smartphone independently.

4.2.5 Conclusions

This study presented a new independent TDS prototype based on an open-source embedded platform which makes it independent of PC/smartphone, visual feedback, and connecting with a wide range of devices in the user environment such as PC, smartphone, TV, IoT

devices, robotic arms, and so forth. Evaluation study by fifteen novice users showed similar performance in playing a 4-command computer game (Fish Tales) in comparison to keyboard arrow keys. Three different wheelchair driving strategies were implemented, tested and compared with the default joystick controller in a constraint driving track. Latched mode of wheelchair driving was found to be the fastest and unlatched mode, the least erroneous among them.

CHAPTER 5

EVALUATION (PARTICIPANTS WITH TETRAPLEGIA)

The evaluation studies involved with the tetraplegic participants to find the usability of mTDS. The following sections will describe both the PC access and wheelchair access study procedure, experiments, and evaluation results.

5.1 Computer Access

The experiment protocol, multimodal tongue drive system, different task descriptions, and performance evaluation parameters are described in the following subsections.

5.1.1 Protocol

The study was approved by the Schulman Institutional Review Board (IRB) at Jacksonville, FL. Fourteen volunteers (all males) with tetraplegia, ages 20-62 years old, naive with respect to the mTDS usage, native English speaker participated in this study. The study involved three sessions per subject, first two sessions were conducted within a week and third session, 7-10 days after the second session. Each session was divided into three parts (preparation, tongue command training, and execution) and limited to 3 hours to reduce fatigue. To be prepared consistently, participants watched an instructional video during the first session, introducing them to mTDS's various input modalities, and all the tasks. The preparation was continued with a small disk-shaped magnetic tracer ($\phi 3.2mm \times 1.6mm$) from K&J Magnetics (Jamison, PA) being attached to the subject's tongue, $\sim$1 cm from the tip, using cyanoacrylic adhesive, after drying the tongue with a paper towel, similar to the previous study [76]. The magnetic tracer attachment in this study was temporary, but at the same location where the tongue piercing should be done for long term use of the mTDS.

5.1.2 Multimodal Tongue Drive System (mTDS)

The multimodal Tongue Drive System (mTDS) utilizes three key remaining abilities (speech, tongue, and head movements) of the people with tetraplegia and convert it to control commands to access a computer [62, 63]. A tiny magnet is attached to the tip of the tongue to capture seven different tongue gestures. When a user moves the tongue, it changes the magnetic field sensed by four magnetometers of the wearable headset shown in . A machine learning algorithm is trained and used to find the real time tongue gestures [77]. Tongue commands are used as discrete switch like inputs such as four keyboard arrow keys and left/right mouse clicks. A inertial measure unit (IMU) is mounted to the top of the headset which captures head movements using a accelerometer and gyroscope. A Kalman filter based sensor fusion algorithm is developed to capture head pitch and roll. Head movements are converted to 2D mouse cursor movements using a transformation model discussed in [63]. Speech is captured using a microphone attached to a arm of the headset and transferred to PC for speech recognition using a commercial tool Dragon Naturally Speaking. In this study, left select and right select tongue commands are used for left and right mouse clicks, head pitch and roll for vertical and horizontal cursor movements, speech recognition for typing similar to the previous study [76].

5.1.3 Center-out Tapping Task

Center-out Tapping Task evaluates the performance of pointing devices using Fitts's law [66]. A round target with different diameters (30, 60 and 122 pixels) and distances (61, 122, 244 pixels) from the center of the screen randomly was shown at a time (Figure 23). Participants were instructed to move the cursor from the center of the screen using their head pitch and roll and click using Left select tongue command. A new target appeared upon selecting the previous one. Each round included 48 random targets. The following metrics are used to evaluate the performance of this task.

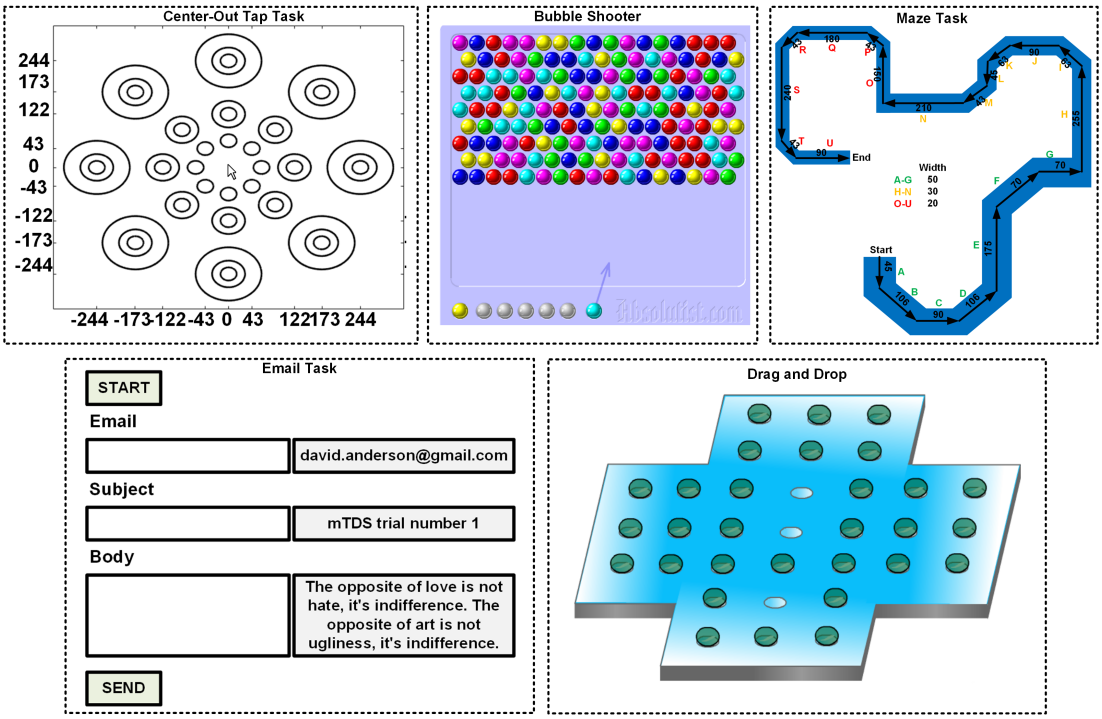

Figure 23: Five tasks are performed by the people with tetraplegia, centerout tapping task, bubble shooter game, maze tasks, sending an email, and drag and drop task. For centerout tapping task, each target was shown at a time. bubble shooter and drag and drop, and maze tasks were limited to 2 mins. Email tasks required typing to each boxes and navigating cursor and clicking to send button.

5.1.3.1 Throughput, $TP_{avg}(bits/s)$

TP_{avg} measures the information transfer rates from a pointing device to PC. It can be found from the ratio between the index of difficulty (ID) and the average movement time (MT) with the same condition [66, 67]. The mean of mean throughput is calculated using Equation (38).

$$TP_{avg} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{m}\sum_{j=1}^{m}\frac{ID_{ij}}{MT_{ij}}), \tag{38}$$

where m is the total number of targets and n is total number of participants. ID is calculated using Shannon's formula shown in Equation (39).

$$ID = \log_2(\frac{D}{W} + 1), \tag{39}$$

where D is the distance from the center of the screen to the center of the target and W is the diameter of the circular target, shown in Figure 23. ID is higher for smaller and farther

targets than larger and closer targets. Higher ID targets are more difficult to reach than the lower ones.

5.1.3.2 Reaction Time, $RT_{avg}(s)$

RT_{avg} finds the time between a target appeared and the participant started to move the cursor towards it [23]. The reaction time is reported as the mean of mean value using Equation (40),

$$RT_{avg} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{m}\sum_{j=1}^{m} RT_{ij}) \tag{40}$$

5.1.3.3 Error Rate, $ER_{avg}(\%)$

ER_{avg} is the percentage of error while selecting a target using mTDS. $ER_{avg}(\%)$ is the ratio of the sum of error selections and the total number of selections calculated using Equation (41) [23].

$$ER_{avg}(\%) = \frac{\sum_{j=1}^{x} Errorselection_j}{\sum_{i=1}^{y} selection_i} \times 100, \tag{41}$$

where x is the total number of selections outside of the target and y is the total number of selections for all subjects in 3 sessions. Here, $Errorselection$ parameter is either '1' or '0' based upon the wrong or correct selection, respectively.

5.1.3.4 Path Efficiency, $PE_{avg}(\%)$

PE_{avg} is the ratio of the euclidean distance from the start point to the target center and the actual distance navigated by the cursor in percentage shown in Equation (42) [23],

$$PE_{avg}(\%) = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{m}\sum_{j=1}^{m} \frac{eucl.distance}{nav.distance} \times 100) \tag{42}$$

5.1.4 Maze Navigation Task

This task evaluates the efficacy of cursor navigation using head movement in a constraint situation as shown in Figure 23. The width of the track (in pixels) is changed to increase task difficulty while cursor approaches towards the endpoint of the maze. The maze widths are varied from 50 to 30, and eventually to 20 pixels. Each maze was divided into 21

segments. The transition from one segment to another required a direction change by $45°, 90°, 135°, 225°$ or $315°$ relative to the previous segment. The duration of the task was limited to 2 mins for each round. Maze layout was randomly selected from 8 patterns with similar difficulty levels (length: 2262 ± 48 *pixels*) to prevent adaptation bias [76].

5.1.4.1 Maze Throughput, $TPm_{avg}(bits/s)$

TPm_{avg} combines both the speed and accuracy of the input device during cursor navigation in a constrained path. The mean of mean maze throughput [73] was reported to quantify subjects' performance which is defined in Equation (43).

$$TPm_{avg} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{t}\sum_{j=1}^{t}(\frac{ID_p P_{in}(1-P_{out})}{CT})_{ij}), \tag{43}$$

where P_{in} and P_{out} are the percentages of distances that the cursor has traversed inside and outside of the maze, respectively [73]. Maze completion time, CT, is the time required for a participant to navigate from start to endpoint, and t is the total number of trials. Path index of difficulty, ID_P in *bits*, is calculated using Shannon's formula from the length (L) to width (W) ratio of the maze segments using Equation (44),

$$ID_p = \sum_{i=1}^{s}\log_2(\frac{L_i}{W_i}+1), \tag{44}$$

where s is the number of maze segments.

5.1.5 Bubble Shooter

Bubble shooter task evaluates sequential use of the mTDS. In this game, participant needs to direct the arrow using the head movement and click using the Left Select tongue command to shoot the bubble. The user interface of this task in shown in Figure 23. Here the participant is targeting a cyan color bubble to shoot towards two cyan bubbles at the top. When user hits three or more same color bubble it pops and increases the score. This task was played for 2 mins.

5.1.5.1 *Shooting Score, SS_{avg}*

The mean of mean shooting score is user as performance metric which is calculated using Equation (45).

$$SS_{avg} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{t}\sum_{j=1}^{t} score_{ij}), \tag{45}$$

where *score* is the score during a trial.

5.1.6 Puzzle

This puzzle task is similar to a peg game. It evaluates the efficacy of using two modalities (tongue and head movements) of mTDS simultaneously. Here, a participant drags a ball using Left tongue command and move it using head. The dragged ball needs to move precisely to the empty spot. Otherwise, it bounces back to the original location. A ball can only be moved over another ball to remove it. Once a dragged ball is moved successfully the middle ball disappeared. This task was time limited to 2 mins.

5.1.6.1 *Drag and Drop (DnD_{avg})*

At the end of the time, a screenshot is taken to count the number of drag and drop which is used as the performance measure. Equation (46) represent the mean of mean drag and drop score (DnD_{avg})

$$DnD_{avg} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{t}\sum_{j=1}^{t} \#\, of\, DD_{ij}), \tag{46}$$

where DD is the total number of drag and drop during a trial.

5.1.7 Sending an Email

This task evaluates the mTDS while doing a complex and fundamental computer access task that utilizes cursor navigation, clicking, and text entry, in a combined or sequential manner. Participants were asked to compose an email with a randomly selected content, chosen from 60 sample texts. The length of each email was 2-3 sentences with comparable length and complexity (24.8 ± 4.7 words). The task was constrained to 5 mins, and the email user interface was customized in a LabVIEW environment. A fixed sender and receiver addresses

were used to minimize inconsistencies among subjects. Figure 23 shows the user interface for this task. The following metrics are used to evaluate the task performance.

5.1.7.1 Task Completion Time, $TCT_{avg}(min)$

TCT_{avg} is the time required (min) to complete the email task, and only captures the subject's speed of finishing this task [76]. It is calculated using Equation (47).

$$TCT_{avg} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{t}\sum_{j=1}^{t}TCT_{ij}), \tag{47}$$

where TCT is the task completion time for a single trial.

5.1.7.2 Text Entry Error Rate, $TEER_{avg}(\%)$

$TEER_{avg}$ is calculated using minimum string distance (MSD) statistics using Equation (48).

$$TEER_{avg}(\%) = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{t}\sum_{j=1}^{t}ErrorRate_{ij}), \tag{48}$$

where $ErrorRate$ is defined as (Equation (49)).

$$ErrorRate = \frac{MSD(A,B)}{max(|A|,|B|)} \times 100\%, \tag{49}$$

where A and B are the characters in string typed and shown to the participants. MSD is calculated using [78]. $max(|A|,|B|)$ is the maximum string length between string A or B.

5.1.7.3 Typing Speed, $TS_{avg}(wpm)$

TS_{avg} finds the mean of mean typed words per minute using Equation (50). It captures the speed of typing during the task. However, it includes both the cursor navigation and clicking time.

$$TS_{avg} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{t}\sum_{j=1}^{t}\frac{words\,typed_{ij}}{TCT_{ij}}), \tag{50}$$

5.2 Results

The results of performance metrics of the five different tasks are presented in the following sections. The performance improvements of each parameter over three sessions are also presented. An analysis is also done to find the statistical significance of the improvements for every task.

5.2.1 Center-out Tapping Task

The overall performance metrics for center-out tapping task is presented in Table 12. All the metrics are presented in terms of mean ± standard error of mean values for first, second, and third session.

The overall mean throughput, TP_{avg} during the first session was 0.85 bps, which improved to 1.11 bps during the third session. Therefore, TP_{avg} was increased 22.6%, 6.6%, and 30.7% from first to second, second to third, first to third session, respectively.

The overall mean reaction time, RT_{avg} for first, second, and third sessions were $0.94s, 0.88s$, and $0.79s$, respectively. There were 6.4%, 10.83%, and 16.53% improvements observed from first to second, second to third, and first to third sessions, respectively.

The overall percentage error rate, ER_{avg} showed a different trend. ER_{avg} during the first, second, and third sessions were 23.41%, 29.49%, and 25.72%, respectively. ER_{avg} was increased by 25.98% from from first to second session, but decreased by 12.80% from second to third session. However, overall ER_{avg} was increased by 9.85% from first to third session. It is expected since the movement time. MT was decreased from first to third session. This outcome is due to the speed vs. accuracy consideration.

Improving cursor navigation path efficiency was not one of the goals of the center-out tapping task. However, it captures how efficiently the cursor was navigated to reach the targets. The overall cursor navigation path efficiency, PE_{avg} were 62.58%, 57.74%, and 60.49% for first, second, and third session, respectively. PE_{avg} decreased by 7.74%, increased by 4.77%, and decreased by 3.34% from first to second, second to third, and first to third session, respectively.

5.2.2 Maze Navigation Task

Overall maze navigation throughput, TPm_{avg} captures both speed and accuracy of this navigation task. TPm_{avg} was $0.31bps, 0.42bps$, and $0.47bps$ for first, second, and third session respectively. TPm_{avg} was improved 34.8%, 12.99%, and 52.31% from first to second, second to third, and first to third session, respectively. It resulted improvements during the consequent sessions. However, the improvements from second to third session was less

Table 12: Performance metrics ($mean \pm sem$) over three sessions

Metrics	Session 1	Session 2	Session 3
Center-out Tapping Task			
$TP_{avg}(bpm)$	0.85 ± 0.01	1.04 ± 0.03	1.11 ± 0.02
$RT_{avg}(s)$	0.94 ± 0.02	0.88 ± 0.02	0.79 ± 0.02
$ER_{avg}(\%)$	23.41 ± 14.16	29.49 ± 16.20	25.72 ± 19.05
$PE_{avg}(\%)$	62.58 ± 0.69	57.74 ± 0.78	60.49 ± 0.79
Email Task			
$TCT_{avg}(min)$	0.87 ± 0.06	0.65 ± 0.02	0.64 ± 0.02
$TEER_{avg}(\%)$	32.41 ± 4.36	17.41 ± 2.63	31.62 ± 4.68
$TS_{avg}(wpm)$	41.32 ± 2.67	55.49 ± 2.52	49.23 ± 3.05
Bubble Shooting Task			
SS_{avg}	766.74 ± 90.54	905.83 ± 70.91	1123.67 ± 126.69
Maze Task			
$TPm_{avg}(bps)$	0.31 ± 0.03	0.42 ± 0.03	0.47 ± 0.04
Drag and Drop Task			
DnD_{avg}	4.43 ± 0.37	6.53 ± 0.52	7.7 ± 0.75

compared to first to second session.

5.2.3 Bubble Shoot Task

The overall shooting score SS_{avg} represents both the speed and accuracy of this task. SS_{avg} during the first, second, and third sessions were 766, 905, and 1123 respectively. The SS_{avg} was improved by 18.14%, 24.05%, and 46.55% from first to second, second to third, and first to third session, respectively. The improvements were consistent during the consecutive sessions.

5.2.4 Puzzle Task

This task required both the speed and accuracy to improve the overall number of drag and drop, DnD_{avg}. The DnD_{avg} for first, second, and third session was $4.43, 6.53$, and 7.7, respectively. The DnD_{avg} was improved 47.4%, 18.01%, and 73.95% from first to second, second to third, and first to third session, respectively. The improvement of second to third session was less compared to first to second session.

5.2.5 Email Task

The overall task completion time, TCT_{avg} for email task of first, second, and third session was $0.87min, 0.65min$, and $0.64min$, respectively. The TCT_{avg} was improved by 26.18% from first to second session, 1.12% from second to third session. However, it improved from first to third session by 27%.

The overall typing speed, TS_{avg} captures only the speed of the task. Here, the TS_{avg} was $41.32wpm, 55.49wpm$, and $49.23wpm$ during first, second, and third session respectively. It improved by 34.28% during first to second, decreased by 11.28% from second to third, and improved by 19.13% from first to third session, respectively. The improvement is less from second to third than first to second session.

The overall text entry error rate, $TEER_{avg}$ presents the accuracy of the task. The $TEER_{avg}$ of first, second, and third session was 32.41%, 17.41%, and 31.62%, respectively. The $TEER_{avg}$ was improved by 46.29%, deteriorated by 81.63%, and improved by 2.45% from first to second, second to third, and first to third session, respectively. The $TEER_{avg}$ improvements were consistent during the consecutive sessions.

5.2.6 Statistical Analysis

A one-tailed paired t-test is done to find the learning ability of all participants with tetraplegia. Table 13 shows the results of the statistical analysis for each performance metric. There is a significant difference between the mean throughput, TP_{avg} from first to last session. Reaction time, RT_{avg}, error rate, ER_{avg}, and path efficiency, PE_{avg} mean differences are not statistically significant.

The mean maze throughput, TPm_{avg} has a significant difference between first to last session which shows that statistically the participants learned over the sessions. The average shooting score, SS_{avg} also has a mean different which shows the learning is statistically significant from first to last session of the experiments. The average number of drag and drop, DnD_{avg} during the puzzle task also has statistically significant difference which shows learning as well.

For the email task, statistical analysis is done for task completion time, TCT_{avg}, typing

Table 13: Statistical Analysis (one-tailed t-test) : Learning Analysis from First to Last Session

	t	t_{cr}	df	p	H_0
Center-out Tapping Task					
TP (bits/s)	2.71	1.83	9	0.012	rejected
RT (s)	1.52	1.83	9	0.082	accepted
ER (%)	-0.07	1.83	9	0.53	accepted
PE (%)	-0.04	1.83	9	0.52	accepted
Maze Task					
TPm (bits/s)	5.52	1.83	9	0.00002	rejected
Game Task					
*SS	3.01	1.83	9	0.007	rejected
Email Task					
TCT(min)	3.03	1.83	9	0.007	rejected
TS(wpm)	2.78	1.83	9	0.011	rejected
TEER(%)	-0.52	1.83	9	0.69	accepted
Puzzle Task					
DnD	3.54	1.86	8	0.0038	rejected

TP: Throughput, RT: Reaction Time, ER: Error Rate, PE: Path Efficiency, TPm: Maze Throughput, SS: Shooting Score, TCT: Task Completion Time, TS: Typing speed, TEER: Text entry error rate, DnD: # of drag and drop, t: one sided t-value, tcr: critical t-value with $\alpha = 0.05$, df: degree of freedom, P: p value, H0: Null hypothesis.*statistically significant

speed, TS_{avg}, and text entry error rate, $TEER_{avg}$. The mean difference for TCT_{avg} and TS_{avg} are statistically significant. However, there is no difference in the mean $TEER_{avg}$ from first to last session. It explains that participants were able to improve the typing speed of the email task but not the accuracy of the typing.

5.2.7 Discussion

The trend of the learning is not plateaued during a three session study. However, a good learning ability and improvements were consistent during all the tasks. During the center-out tapping task, significant improvements were not seen in the reaction time, error rate, and path efficiency. However, throughput is improved significantly over the sessions. More session data would reveal this results with more confidence. Text entry error rate is not improved significantly. However, the typing speed improvement is statistically significant. This

might be due to the speech recognition accuracy of the Dragon Naturally Speaking. Based on the results, a multi-session (until performance plateaued) study of mTDS is suggested to study the learning ability of the device with more accuracy.

5.2.8 Conclusion

This work shows the feasibility of using a multimodal assistive technology in the population with tetraplegia. During a computer access, statistically significant improvements were found in center-out tapping, bubble shooting, and maze navigation tasks. Puzzle task shows the efficacy of mTDS while integrating multiple remaining abilities, here tongue and head movements to do complex computer interaction like drag and drop. Significant improvements were found in this task over three sessions of usage. A compound task such as sending en email was also done in this study. Participants task completion time and typing speed were improved significantly over the sessions. However, typing accuracy improvement was not significant.

5.3 Wheelchair Driving

The purpose of this study is to investigate the performance of the mTDS which uses 2 different modalities (tongue movements and head movements) to control a wheelchair. The participants will be asked to navigate a wheelchair in a standardized course (Figure 22) using mTDS by combining their head and tongue movements, as well as their own drive method. The efficacy of the mTDS will be evaluated using the speed and accuracy of the driving.

5.3.1 Protocol

Target recruitment for this study is 30 participants age: 18 to 74, tetraplegia due to spinal chord injury or neurological disorder. A magnet will be attached to their tongue similar to discussed in Chapter 4. As preparation, they will go through a waveform check, calibration, and training using the touchscreen display as described in Chapter 2. A testing will also be done to check the proper verify the machine learning training quality. After functional testing, the device is ready to drive a wheelchair. The wheelchair user interface is shown in

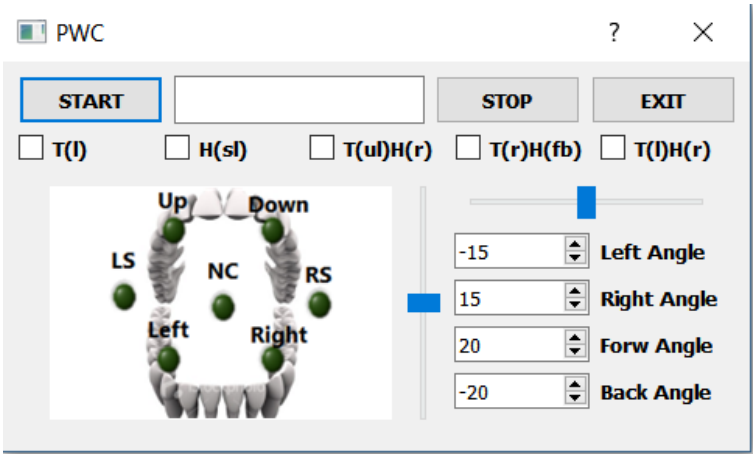

Figure 24: User interface to drive a wheelchair using head only, tongue only, and tongue-head combined movements.

Figure 24.

The mode of driving will be randomized before starting the trial. The tongue Only, head Only, Head/Tongue Combination #1, Head/Tongue Combination #2 modes are explained in the following sections.

5.3.2 Head Only Mode

The Head Only Mode is implemented to compare the tongue motion based driving vs. the head motion based driving. Head movement is incorporated to move the PWC forward/reverse and left/right. Forward flexion of the head moves the WC forward while head extension moves the WC in reverse direction. Side bending the head to left and right will turn the WC in left or right directions respectively. The user can combine the forward/reverse and left/right head movements to drive the WC in forward left/right or reverse left/right directions.

5.3.3 Tongue Only Mode

Forward, Backward, Left, and Right tongue commands moves the WC in the forward, reverse, left, and right directions, respectively. Upon holding one of these four commands, the amplitude of the driving vector increases thus moving the WC faster. In this study, the mTDS uses the latched mode of tongue-controlled WC driving. The user will assign a Left Select command to latch the WC (similar to setting up cruise controlled driving) in

a forward driving mode. While being in the forward mode, the user can assign left/right commands resulting forward left/right movements of the WC. In this mode, only the forward command is latched. However, during the unlatched condition (if necessary in space constraint part of the driving track), the user can assign four tongue command to move the WC to an individual direction (forward/ backward/ left/ right).

5.3.4 Head/Tongue Combination #1

In this mode of driving, both tongue and head movements are combined to drive the WC. Forward/reverse movement of the WC is controlled using tongue commands and the turning or steering is controlled by the head movements. The WC can be latched using Left Select tongue command. If latched, head movement is used to steer left/right. In this mode, the WC can steer left/right in reverse since tongue and head movements can be assigned simultaneously.

5.3.5 Head/Tongue Combination #2

In this mode of driving, forward/reverse movement of the WC is controlled using head movement and the turning or steering is controlled by the Left/Right tongue commands. The latched mode is activated using tilting the head forward above a threshold and unlatched using a left head rotation gesture. While latched, left/right tongue commands are used to steer the WC in left/right direction. The same can be done to move the WC to reverse left/right by tilting the head backward.

5.3.6 Performance Measurement

Two performance metrics will be captured during the WC driving completion time and driving error as described in Chapter 4. Completion time captures the speed of the WC task and driving error captures the accuracy of the task. The overall mean of mean metrics will be reported to find the better driving strategy in terms of speed and accuracy.

5.3.7 Conclusion

This work shows the feasibility of using a multimodal assistive technology in the population with tetraplegia. Two studies were planned to measure the efficacy and learning capability

of the mTDS. A computer access study was completed. A wheelchair study is planned to evaluate the efficacy of mTDS during the mobility task. Five different modes: head only, tongue only, head tongue combination #1, and head tongue combination #2 were implemented for this study. These modes of driving will be compared with the participants default mode of driving and the results will be reported in terms of task completion time and driving error.

CHAPTER 6

FUTURE RESEARCH

6.1 Speech Recognition in mTDS

The mTDS designed in this work captures speech using a microphone and rely on a computer/ smartphone to process the speech for the purpose of typing. However, it is possible to process some keywords in the system using a lightweight supervised speech recognition algorithm. This algorithm can be implemented in the mTDS control unit. The user will be able to choose between tongue and speech commands to drive a wheelchair or access a PC.

6.2 Typing: mTDS

The accuracy of typing using speech recognition (SR) is not satisfactory while being in a noisy environment. It is also not feasible or preferable to use SR in special scenarios, such as typing a password or a private message while being in a crowd. On the other hand, using only six tongue commands and an on-screen keyboard to type long sentences is also difficult, even with word prediction. We intend to use a combination of the tongue and head motion to type more efficiently in terms of both speed and accuracy. Figure 25 shows a proposed keyboard layout, where pitch /roll of the head motion will be used to select each block of characters to type one of the 6 characters in that block by tongue commands. It will facilitate selection of up to 54 characters by combining the two mTDS inputs simultaneously. Pitch/roll can be divided into three ranges. First range, which will be within the normal range of head movements, will be defined as no command zone, second range will select the lighter colored blocks, and third range will select the deeper colored blocks, shown in Figure 25. Frequently used letters will be typed directly with only six tongue commands, as shown in the middle of Figure 25. Once implemented with an effective, intuitive, and clear UI, this typing technique will be compared with existing methods, such as onscreen, EdgeWrite [79], T9, and standard keyboards by able-bodied participants.

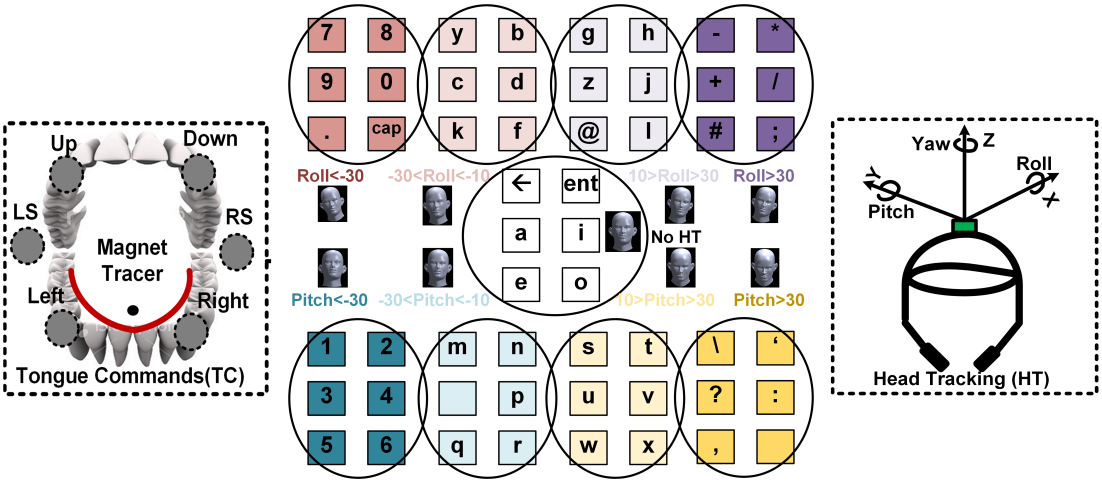

Figure 25: Proposed keyboard layout by combining head motion and tongue commands. Lighter colors represent less head pitch/roll.

6.3 Evaluation: Smartphone

The evaluation of mTDS to access a smartphone is not done so far in this work. Although it is very similar to control a PC, but a evaluation study with the end user while doing various smartphone tasks such as making a phone call, testing, web surfing, and so forth is necessary. This type of evaluation study can a part of future studies with end users.

6.4 Evaluation: Long Term Study

The end user study involved only three sessions while accessing a PC in past. A good learning curve was observed in most of the performance metrics. However, those metrics did not reach to saturation. In other words, there was a scope to improve those parameters further with more learning/usage. A long term end user study will be able to provide a better conclusion regarding those concerns. This type of study can also answer, how close a person with tetraplegia can perform using mTDS in comparison to a person using a standard input device such as keyboard-mouse combination while doing the same task (web surfing or sending an email).

6.5 Noninvasive Device Design

The mTDS requires to attach a magnet on the tongue to capture tongue gestures. However, other muscle signal/facial expression can be captured and used as a discrete type of inputs

without requiring any instrumentation on the tongue. This will encourage more people without disability to use this device for various other application such as playing a game using mTDS or doing multitasking more efficiently.

6.6 Others

The application of mTDS is not explored completely in this work. However, it is possible to use mTDS for robotic rehabilitation [60], silent speech recognition [80], speech therapy [53], and various other applications. The application of the mTDS is encouraged to explored highly since this device can capture and process multiple remaining abilities simultaneously. The application of mTDS for physical therapy such as neck exercise can be explored in future.

CHAPTER 7

CONCLUSIONS

In this work, a novel multimodal assistive technology is developed by leveraging three key remaining abilities such as speech, tongue, and head movements of the people with tetraplegia to provide them efficient access to computers, smartphones, and wheelchairs. The mTDS can process all three modalities real-time which provides end-user flexibility to use them sequentially or simultaneously based on the task nature. Efficient algorithms to process tongue and head gestures are developed and implemented in the wearable headset. The optimized tongue gesture processing algorithm resulted high accuracy outcome for both robot emulated and 15 able-bodied human subjects' data. Tongue and head gesture processing algorithms made the mTDS truly standalone to interface with various devices without a need of a processing unit (PC or smartphone) to control devices such as smartphones, computers, wheelchairs, smart-homes and so forth.

Further, a processing unit is also designed which include a touch-screen based visual feedback to train, calibrate, and switch between the interface devices. The processing unit includes various wireless connectivity and interface ability to drive a wheelchair or control a smart-home appliances. Several driving techniques are proposed and implemented in the processing unit using only tongue, head, and tongue-head combination. In a able-bodied subject study, three different wheelchair driving strategies using tongue were tested and compared with the joystick controller in a constraint driving track. Latched mode wheelchair driving strategy was found to be the fastest and unlatched mode, the least erroneous among them.

A multitask computer study with able-bodied participants showed that the discrete single mode, TDS generates higher throughput while doing a center-out tapping task compared to mTDS. However, using the mTDS users average error rate and target reaching capabilities

improved significantly. Participants resulted better outcome during a constraint cursor navigation using tongue as opposed to head. However, during a unconstrained cursor navigation (playing a game), the mTDS resulted significantly better outcome. The mTDS performed very similar to a keyboard-mouse combination while performing a complex combined task such as sending an email.

During a multitask computer task evaluation study with end-users, mTDS showed consistent improvements while doing all five different computer access tasks. A wheelchair driving study with tongue-head combination, tongue, and head only is underway to evaluate the efficacy of the mTDS usage. The mTDS is expected to provide better efficacy and capabilities to a person with tetraplegia while accessing a computer and driving a wheelchair compared to existing assistive technologies and reach to a similar performance that can be done using standard input devices such as keyboard-mouse combination or joystick.

REFERENCES

[1] "Spinal cord injury facts and figures at a glance." https://msktc.org/lib/docs/Data_Sheets_/SCIMS_Facts_and_Figures_2017_FINAL.pdf. Accessed: 2018-09-06.

[2] "Spinal cord injury facts & statistics." http://www.sci-info-pages.com/facts.html. Accessed: 2018-09-06.

[3] "Costs of living with spinal cord injury." https://www.christopherreeve.org/living-with-paralysis/costs-and-insurance/costs-of-living-with-spinal-cord-injury. Accessed: 2018-09-06.

[4] J. Long, Y. Li, H. Wang, T. Yu, J. Pan, and F. Li, "A hybrid brain computer interface to control the direction and speed of a simulated or real wheelchair," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, no. 5, pp. 720–729, 2012.

[5] T. Blakely, K. J. Miller, S. P. Zanos, R. P. Rao, and J. G. Ojemann, "Robust, long-term control of an electrocorticographic brain-computer interface with fixed parameters," *Neurosurgical focus*, vol. 27, no. 1, p. E13, 2009.

[6] J.-S. Han, Z. Z. Bien, D.-J. Kim, H.-E. Lee, and J.-S. Kim, "Human-machine interface for wheelchair control with emg and its evaluation," in *Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE*, vol. 2, pp. 1602–1605, IEEE, 2003.

[7] A. Bulling and H. Gellersen, "Toward mobile eye-based human-computer interaction," *IEEE Pervasive Computing*, no. 4, pp. 8–12, 2010.

[8] H. Christensen, I. Casanueva, S. Cunningham, P. Green, and T. Hain, "homeservice: Voice-enabled assistive technology in the home using cloud-based automatic speech recognition," in *Proceedings of the Fourth Workshop on Speech and Language Processing for Assistive Technologies*, pp. 29–34, 2013.

[9] J. Musić, M. Cecić, and M. Bonković, "Testing inertial sensor performance as hands-free human-computer interface," *WSEAS Trans. Comput.*, vol. 8, no. 4, pp. 715–724, 2009.

[10] A. Plotkin, L. Sela, A. Weissbrod, R. Kahana, L. Haviv, Y. Yeshurun, N. Soroker, and N. Sobel, "Sniffing enables communication and environmental control for the severely disabled," *Proceedings of the National Academy of Sciences*, vol. 107, no. 32, pp. 14413–14418, 2010.

[11] A. Escobedo, A. Spalanzani, and C. Laugier, "Multimodal control of a robotic wheelchair: Using contextual information for usability improvement," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 4262–4267, IEEE, 2013.

[12] M. M. Moore, "Real-world applications for brain-computer interface technology," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 11, no. 2, pp. 162–165, 2003.

[13] J. P. Donoghue, "Bridging the brain to the world: a perspective on neural interface systems," *Neuron*, vol. 60, no. 3, pp. 511–521, 2008.

[14] M. R. Ahsan, M. I. Ibrahimy, O. O. Khalifa, *et al.*, "Emg signal classification for human computer interaction: a review," *European Journal of Scientific Research*, vol. 33, no. 3, pp. 480–501, 2009.

[15] M. Ghovanloo, M. N. Sahadat, Z. Zhang, F. Kong, and N. Sebkhi, "Tapping into tongue motion to substitute or augment upper limbs," in *Micro-and Nanotechnology Sensors, Systems, and Applications IX*, vol. 10194, p. 1019413, International Society for Optics and Photonics, 2017.

[16] M. Ghovanloo, "Tongue operated assistive technologies," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pp. 4376–4379, IEEE, 2007.

[17] "Tongue touch keypad, newabilities systems inc, santa clara, ca." `www.newabilities.com`. Accessed: 2018-09-06.

[18] L. N. A. Struijk, "A tongue based control for disabled people," in *International Conference on Computers for Handicapped Persons*, pp. 913–918, Springer, 2006.

[19] W. Nutt, C. Arlanch, S. Nigg, and G. Staufert, "Tongue-mouse for quadriplegics," *Journal of Micromechanics and Microengineering*, vol. 8, no. 2, p. 155, 1998.

[20] C. Salem and S. Zhai, "An isometric tongue pointing device," in *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pp. 538–539, ACM, 1997.

[21] "Jouse3." `http://www.jouse.com/`. Accessed: 2018-09-06.

[22] T. S. Saponas, D. Kelly, B. A. Parviz, and D. S. Tan, "Optically sensing tongue gestures for computer input," in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pp. 177–180, ACM, 2009.

[23] J. Kim, H. Park, J. Bruce, E. Sutton, D. Rowles, D. Pucci, J. Holbrook, J. Minocha, B. Nardone, D. West, *et al.*, "The tongue enables computer and wheelchair control for people with spinal cord injury," *Science translational medicine*, vol. 5, no. 213, pp. 213ra166–213ra166, 2013.

[24] X. Huo, H. Park, J. Kim, and M. Ghovanloo, "A dual-mode human computer interface combining speech and tongue motion for people with severe disabilities," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 21, no. 6, pp. 979–991, 2013.

[25] C. Guger, S. Daban, E. Sellers, C. Holzner, G. Krausz, R. Carabalona, F. Gramatica, and G. Edlinger, "How many people are able to control a p300-based brain–computer interface (bci)?," *Neuroscience letters*, vol. 462, no. 1, pp. 94–98, 2009.

[26] N. M. Lopez, E. Orosco, E. Perez, S. Bajinay, R. Zanetti, M. E. Valentinuzzi, and G. de Tecnología Médica, "Hybrid human-machine interface to mouse control for severely disabled people," *system*, vol. 4, no. 11, 2015.

[27] A. Królak and P. Strumiłło, "Eye-blink detection system for human–computer interaction," *Universal Access in the Information Society*, vol. 11, no. 4, pp. 409–419, 2012.

[28] J. P. Hansen, K. Tørning, A. S. Johansen, K. Itoh, and H. Aoki, "Gaze typing compared with input by head and hand," in *Proceedings of the 2004 symposium on Eye tracking research & applications*, pp. 131–138, ACM, 2004.

[29] X. Huo, J. Wang, and M. Ghovanloo, "Introduction and preliminary evaluation of the tongue drive system: Wireless tongue-operated assistive technology for people with little or no upper-limb function.," *Journal of Rehabilitation Research & Development*, vol. 45, no. 6, 2008.

[30] D. Fortune, J. E. Ortiz, and H. N. Tran, "Tongue activated communications controller," June 4 1996. US Patent 5,523,745.

[31] F. Kong, C. Qi, H. Lee, G. D. Durgin, and M. Ghovanloo, "Antennas for intraoral tongue drive system at 2.4 ghz: Design, characterization, and comparison," *IEEE Transactions on Microwave Theory and Techniques*, vol. 66, no. 5, pp. 2546–2555, 2018.

[32] C.-H. Shih, C.-J. Shih, and C.-T. Shih, "Assisting people with multiple disabilities by actively keeping the head in an upright position with a nintendo wii remote controller through the control of an environmental stimulation," *Research in Developmental Disabilities*, vol. 32, no. 5, pp. 2005–2010, 2011.

[33] E.-J. Rechy-Ramirez, H. Hu, and K. McDonald-Maier, "Head movements based control of an intelligent wheelchair in an indoor environment," in *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pp. 1464–1469, IEEE, 2012.

[34] D. Rudi, I. Giannopoulos, P. Kiefer, C. Peier, and M. Raubal, "Interacting with maps on optical head-mounted displays," in *Proceedings of the 2016 Symposium on Spatial User Interaction*, pp. 3–12, ACM, 2016.

[35] M. T. Qadri and S. A. Ahmed, "Voice controlled wheelchair using dsk tms320c6711," in *Signal Acquisition and Processing, 2009. ICSAP 2009. International Conference on*, pp. 217–220, IEEE, 2009.

[36] A. Škraba, A. Kološvari, D. Kofjač, and R. Stojanović, "Prototype of speech controlled cloud based wheelchair platform for disabled persons," in *Embedded Computing (MECO) 3rd Mediterranean Conference on*, pp. 162–165, 2014.

[37] B. Yousefi, X. Huo, E. Veledar, and M. Ghovanloo, "Quantitative and comparative assessment of learning in a tongue-operated computer input device," *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 5, pp. 747–757, 2011.

[38] M. Turk, "Multimodal interaction: A review," *Pattern Recognition Letters*, vol. 36, pp. 189–195, 2014.

[39] L. N. A. Struijk, E. R. Lontis, B. Bentsen, H. V. Christensen, H. A. Caltenco, and M. E. Lund, "Fully integrated wireless inductive tongue computer interface for disabled people," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pp. 547–550, IEEE, 2009.

[40] E. R. Lontis, M. E. Lund, H. V. Christensen, B. Bentsen, M. Gaihede, H. A. Caltenco, and L. N. A. Struijk, "Clinical evaluation of wireless inductive tongue computer interface for control of computers and assistive devices," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pp. 3365–3368, IEEE, 2010.

[41] M. E. Lund, H. V. Christiensen, H. A. Caltenco, E. R. Lontis, B. Bentsen, and L. N. A. Struijk, "Inductive tongue control of powered wheelchairs," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pp. 3361–3364, IEEE, 2010.

[42] L. NS Andreasen Struijk, E. R. Lontis, M. Gaihede, H. A. Caltenco, M. E. Lund, H. Schioeler, and B. Bentsen, "Development and functional demonstration of a wireless

intraoral inductive tongue computer interface for severely disabled persons," *Disability and Rehabilitation: Assistive Technology*, vol. 12, no. 6, pp. 631–640, 2017.

[43] X. Huo, J. Wang, and M. Ghovanloo, "A magneto-inductive sensor based wireless tongue-computer interface," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 16, no. 5, pp. 497–504, 2008.

[44] J. Kim, X. Huo, and M. Ghovanloo, "Wireless control of smartphones with tongue motion using tongue drive assistive technology," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pp. 5250–5253, IEEE, 2010.

[45] X. Huo, J. Wang, and M. Ghovanloo, "A wireless tongue-computer interface using stereo differential magnetic field measurement," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pp. 5723–5726, IEEE, 2007.

[46] H. Park and M. Ghovanloo, "A wireless intraoral tongue–computer interface," *Wireless Medical Systems and Algorithms: Design and Applications*, vol. 56, p. 63, 2016.

[47] J. Kim, X. Huo, J. Minocha, J. Holbrook, A. Laumann, and M. Ghovanloo, "Evaluation of a smartphone platform as a wireless interface between tongue drive system and electric-powered wheelchairs," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 6, pp. 1787–1796, 2012.

[48] "Quantum rehab.." `http://www.pridemobility.com/pdf/brochures/jazzy/quantum_6000.pdf`. Accessed: 2018-09-06.

[49] R. Wallace, "Antenna selection guide," *Texas Instruments, Application Note AN058*, 2009.

[50] "Pride mobility." `http://www.pridemobility.com/includes/download.php?f=Owners_Manuals/US_Jazzy/US_Q6000_Series_om.pdf`. Accessed: 2018-09-06.

[51] E. B. Sadeghian, X. Huo, and M. Ghovanloo, "Command detection and classification in tongue drive assistive technology," in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pp. 5465–5468, IEEE, 2011.

[52] M. N. Sahadat, S. Dighe, F. Islam, and M. Ghovanloo, "An independent tongue-operated assistive system for both access and mobility," *IEEE Sensors Journal*, vol. 18, no. 22, pp. 9401–9409, 2018.

[53] N. Sebkhi, D. Desai, M. Islam, J. Lu, K. Wilson, and M. Ghovanloo, "Multimodal speech capture system for speech rehabilitation and learning," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 11, pp. 2639–2649, 2017.

[54] K.-Y. Chen, S. N. Patel, and S. Keller, "Finexus: Tracking precise motions of multiple fingertips using magnetic sensing," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 1504–1514, ACM, 2016.

[55] A. Jafari, N. Buswell, M. Ghovanloo, and T. Mohsenin, "A low-power wearable stand-alone tongue drive system for people with severe disabilities," *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 1, pp. 58–67, 2018.

[56] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[57] F. Abyarjoo, A. Barreto, J. Cofino, and F. R. Ortega, "Implementing a sensor fusion algorithm for 3d orientation detection with inertial/magnetic sensors," in *Innovations and advances in computing, informatics, systems sciences, networking and engineering*, pp. 305–310, Springer, 2015.

[58] R. Faragher *et al.*, "Understanding the basis of the kalman filter via a simple and intuitive derivation," *IEEE Signal processing magazine*, vol. 29, no. 5, pp. 128–132, 2012.

[59] F. Kong, M. Sahadat, and M. Ghovanloo, "Development and preliminary assessment of an arch-shaped standalone intraoral tongue drive system for people with tetraplegia,"

in *Biomedical Circuits and Systems Conference (BioCAS), 2018 IEEE*, pp. 1–4, IEEE, 2018.

[60] Z. Zhang, S. Ostadabbas, M. Sahadat, N. Sebkhi, D. Wu, A. Butler, and M. Ghovanloo, "Enhancements of a tongue-operated robotic rehabilitation system," in *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*, pp. 1–4, IEEE, 2015.

[61] "Dragon naturallyspeaking." `www.nuance.com/Dragon`. Accessed: 2018-09-06.

[62] M. Sahadat, A. Alreja, P. Srikrishnan, and M. Ghovanloo, "A multimodal human computer interface combining head movement, speech and tongue motion for people with severe disabilities," in *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*, pp. 1–4, IEEE, 2015.

[63] M. N. Sahadat, A. Alreja, and M. Ghovanloo, "Simultaneous multimodal pc access for people with disabilities by integrating head tracking, speech recognition, and tongue motion," *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 1, pp. 192–201, 2018.

[64] M. Sahadat, N. Sebkhi, F. Kong, and M. Ghovanloo, "Standalone assistive system to employ multiple remaining abilities in people with tetraplegia," in *Biomedical Circuits and Systems Conference (BioCAS), 2018 IEEE*, pp. 1–4, IEEE, 2018 (accepted).

[65] "Click-n-type, on-screen keyboard." `http://cnt.lakefolks.org/`. Accessed: 2018-09-06.

[66] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement.," *Journal of experimental psychology*, vol. 47, no. 6, p. 381, 1954.

[67] R. W. Soukoreff and I. S. MacKenzie, "Towards a standard for pointing device evaluation, perspectives on 27 years of fittsâĂŹ law research in hci," *International journal of human-computer studies*, vol. 61, no. 6, pp. 751–789, 2004.

[68] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[69] B. Yousefi, X. Huo, J. Kim, E. Veledar, and M. Ghovanloo, "Quantitative and comparative assessment of learning in a tongue-operated computer input device—part ii: Navigation tasks," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 4, pp. 633–643, 2012.

[70] I. S. MacKenzie and P. Isokoski, "Fitts' throughput and the speed-accuracy tradeoff," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1633–1636, ACM, 2008.

[71] J. O. Wobbrock, E. Cutrell, S. Harada, and I. S. MacKenzie, "An error model for pointing based on fitts' law," in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1613–1622, ACM, 2008.

[72] J. O. Wobbrock, K. Shinohara, and A. Jansen, "The effects of task dimensionality, endpoint deviation, throughput calculation, and experiment design on pointing measures and models," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1639–1648, ACM, 2011.

[73] H. A. Caltenco, B. Breidegard, and L. N. Andreasen Struijk, "On the tip of the tongue: Learning typing and pointing with an intra-oral computer interface," *Disability and Rehabilitation: Assistive Technology*, vol. 9, no. 4, pp. 307–317, 2014.

[74] D. Natapov, S. J. Castellucci, and I. S. MacKenzie, "Iso 9241-9 evaluation of video game controllers," in *Proceedings of Graphics Interface 2009*, pp. 223–230, Canadian Information Processing Society, 2009.

[75] D. C. Hoaglin and B. Iglewicz, "Fine-tuning some resistant rules for outlier labeling," *Journal of the American Statistical Association*, vol. 82, no. 400, pp. 1147–1149, 1987.

[76] M. N. Sahadat, A. Alreja, N. Mikail, and M. Ghovanloo, "Comparing the use of single versus multiple combined abilities in conducting complex computer tasks hands-free,"

IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 26, no. 9, pp. 1868–1877, 2018.

[77] M. N. Sahadat, N. Sebkhi, D. Anderson, and M. Ghovanloo, "Optimization of tongue gesture processing algorithm for standalone multimodal tongue drive system," *IEEE Sensors Journal*, 2018.

[78] R. W. Soukoreff and I. S. MacKenzie, "Measuring errors in text entry tasks: an application of the levenshtein string distance statistic," in *CHI'01 extended abstracts on Human factors in computing systems*, pp. 319–320, ACM, 2001.

[79] J. O. Wobbrock, B. A. Myers, and J. A. Kembel, "Edgewrite: a stylus-based text entry method designed for high accuracy and stability of motion," in *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pp. 61–70, ACM, 2003.

[80] H. Sahni, A. Bedri, G. Reyes, P. Thukral, Z. Guo, T. Starner, and M. Ghovanloo, "The tongue and ear interface: a wearable system for silent speech recognition," in *Proceedings of the 2014 ACM International Symposium on Wearable Computers*, pp. 47–54, ACM, 2014.